Improved Global Routing through Congestion Estimation

Raia T. Hadsell and Patrick H. Madden
SUNY Binghamton CSD Box 6000 Binghamton NY 13902
raia@math.binghamton.edu pmadden@cs.binghamton.edu
http://vlsicad.cs.binghamton.edu

ABSTRACT

In this paper, we present a new method to improve global routing results. By using an amplified congestion estimate to influence a rip-up and reroute approach, we obtain substantial reductions in total congestion. In comparisons with a recently published tool on publicly available benchmarks, our new router is roughly twice as fast, obtains 15.1% reductions in total wire length, and 65.2% reductions in the number of overcongested graph edges. A direct implementation of an old approach also performs extremely well, indicating that some known techniques have been overlooked.

Categories and Subject Descriptors

J.6 [**Computer-Aided Engineering**]: CAD

General Terms

Algorithms

Keywords

Global routing, Steiner trees, congestion

1. INTRODUCTION

As designs have become larger and more congested, global routing has become more difficult. Routing can fail to complete, or can take an unacceptably long time; [14] discusses these issues in the context of routability-driven placement. Optimization of congestion in global routing problems is NP-hard; thus, it is unlikely that there can be any efficient algorithm that guarantees success. In this paper, we extend a well known heuristic approach, improving the quality of global routing solutions substantially, with minimal impact to run time.

In this paper, we present the *Chi Dispersion Global Routing* tool, which extends an earlier graph-based global router[8]. We consider over-the-cell routing in this paper, using the grid-based model that has been used in recent works (for example [6, 1, 12]). Using the benchmarks provided by the authors of [6], we are able to obtain average wire length reductions of 15.1%, and a 65.2% reduction in an "over-congestion" metric. In addition to improved routing quality, our new tool is also substantially faster.

Our approach to the routing problem utilizes an earlier method by Linsker for printed circuit board (PCB) routing[11]; this work has apparently been overlooked by many in the modern VLSI IC routing community. We enhance this method by introducing a feedback loop to each rip-up and reroute iteration, in order to inform the maze router of congested areas that should be avoided. The congested areas are determined in two ways: static estimates about congestion are given by a probabilistic algorithm, and dynamic information about congestion is obtained from previous iterations of the rip-up and reroute process. The composite information is used to introduce an artificial weight to the congested edges in the graph, causing subsequent routes to disperse from these areas during reroute.

Our primary contributions are the following. (1) We highlight previous research results from PCB routing, showing that they impact modern VLSI IC routing. A direct implementation of [11] in fact outperforms more recent tools by a wide margin. (2) We present methods to weight routing graphs using congestion estimates; this results in significantly improved routing results. (3) We develop an *amplification* approach that allows the congestion estimates to impact the most dense regions, while avoiding the introduction of detours in less congested regions.

The remainder of this paper is organized as follows; we first describe the routing model in more detail, and survey recent work in the area. We next describe our routing approach, which enhances an earlier rip-up and reroute technique with congestion estimation. We then compare our tool to recently published tools on widely available benchmarks; the improvements are surprisingly large. The paper concludes with a summary of results, and a consideration of future work.

2. PREVIOUS WORK

For many years, a channel-based global routing model was appropriate. With limited numbers of routing layers, all connections were restricted to the area between standard cell rows, or around macro blocks. Global routers (for example [15]) assigned nets to these routing channels, and inserted feedthrough space in cell rows; this is illustrated in Figure 1(a). With added numbers of routing layers, the space used for channel routing can be eliminated, and aggressive over-the-cell routing is now popular. This is illustrated in Figure 1(b). Global routing tools using this formulation include Labyrinth[6] and the multi-commodity flow based router of Albrecht[1].

In the modern routing model, the problem is normally formulated as a variant of multi-commodity flow. We are given a graph $G(V,E)$, where each edge e_{ij} between vertices v_i and v_j has a ca-

DAC'03, June 2–6,2003,Anaheim,California,USA.

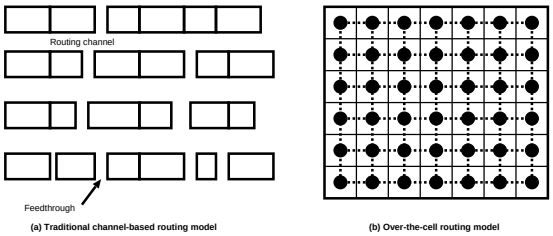

Figure 1: A traditional channel-based routing model, and the current over-the-cell routing model. The vertices in the over-the-cell model represent rectilinear regions, while edges in this model represent the borders between regions.

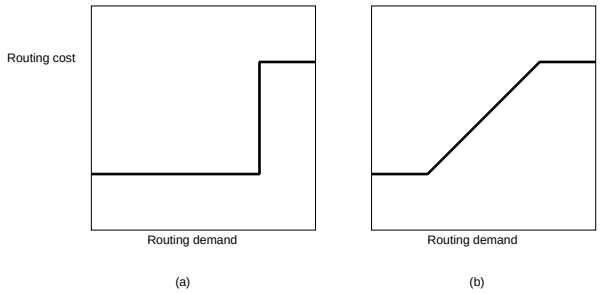

Figure 2: In rip-up and reroute approaches, routing cost is a function of the demand for graph edges. In many recent works, cost increases abruptly when demand reaches (or approaches) capacity (a); in the routing tool by Linsker, routing cost is a linear function of demand (b).

pacity c_{ij}. We must form Steiner trees to connect subsets of vertices, such that the capacity constraints are obeyed, and we minimize total tree length. Normally, rectangular "global cell" or "GCell" regions are defined over the circuit area, and these regions are the vertices of a routing graph. The borders between regions are represented by edges; as there is finite routing space on the borders, each edge has a *capacity*.

In [14], it was observed that detail routing becomes difficult when routing demand approaches physical capacity; if demand exceeds capacity, failure is assured. Thus, the *capacity* of an edge in a global routing graph is generally tuned towards levels where successful detail routing is likely. A routing solution which exceeds capacity is said to be *over congested*, but may still be routable. A common optimization objective in global routing is to minimize both total wire length, and the number of over-congested graph edges.

Many global routing tools utilize rip-up and reroute; this is a well-known and popular technique. Generally, the approach begins with decomposition of multipin nets into pairs of pins (using either Spanning Tree algorithms[13, 9] or Steiner Tree heuristics[5, 2]). Following the decomposition, each tree edge is routed using a maze router. Normally, Dijkstra's algorithm[4] is used, with the routing cost for each edge or vertex in the graph being adjusted when the usage of the resource increases.

The global router by Albrecht[1] utilizes a new multi-commodity flow approximation algorithm, to obtain results that can be proven to be close to optimal. The approximation algorithm uses fractional flows; thus, it is necessary to perform randomized rounding, with traditional rip-up and reroute to complete the process.

The formulation for over-the-cell VLSI IC global routing is in fact surprisingly close to a formulation used for PCB routing. In [11], a global routing tool which used a formulation as described above was presented. A key contribution of this work was the study of routing cost functions, to allow consideration of routing congestion. We use these results in our work, and describe this in more detail in the following sections.

3. GLOBAL ROUTING WITH CONGESTION ESTIMATION

Our routing approach extends previous work through the introduction of a congestion estimation method; this influences the routing of individual connections, improving the solution quality. In this section, we first describe the routing method of [11] in more detail. We then present our method of predicting congestion (which is similar to techniques found in some placement tools). The routing costs are influenced by the congestion estimates; our objective is to *disperse* routes from areas we expect to be congested.

Basic Routing Approach

The basic approach in our routing tool is well known; we follow the general outline of Linsker[11]. We first decompose multi-pin nets into sets of point to point connections, and then route each connection.

In our work, we use the Steiner tree heuristic of Borah, Owen, and Irwin[2]. This heuristic obtains good quality trees (close to optimal on random problems), and also has low complexity; our implementation is $O(n^2)$. We will refer to edges of an interconnect tree as *wires*, to avoid confusion with the edges of the routing graph.

Each wire is routed using a maze router; we have implemented Dijkstra's algorithm, taking care to ensure that our implementation is efficient. We rip-up and reroute each edge, in order, for a number of iterations; we find that solution quality converges quickly, and that large numbers of iterations seldom improve quality.

As is done in most global routing tools, we monitor the number of routes using any particular edge in our routing graph, and use this to adjust routing cost. We consider this to be an extremely important issue in the construction of an effective global routing tool. In [11], it was observed that having routing cost increase linearly with congestion was far more effective than having a "step" cost function; these observations were confirmed in [3]. Despite this, several recent global routers (for example, Labyrinth[7], and the global router used for reference by Albrecht[1]), have routing cost which increases abruptly when the number of routes on a graph edge reaches the edge capacity. The routing cost functions are illustrated in Figure 2.

In [7], route cost is defined as $\alpha \times overflow + length$; the route is penalized only if it exceeds the routing capacity. By contrast, we assign unit cost to an edge until it reaches 80% of capacity, and increase cost linearly (to a maximum cost of 10) until it reaches 40% above capacity. [3] used a similar technique, but dynamically updated routing cost because no capacity constraints were included with their benchmarks.

The intuition for the "linear" cost function is clear: early routes will encounter an uncongested routing graph, and will utilize the resources without consideration for the demands of later routes. Later routes will then detour around congested regions, increasing the overall routing demand considerably. During rip-up and reroute, the connections routed in the early stages will have difficulty in being rerouted, because of the added wire length.

Congestion Estimation and Amplification

Prior to initial routing, the method of Linsker has no indication as to where congestion is to be expected; thus, the first routes may pass through areas that will have high demand, even when there

are viable alternatives. During rip-up and reroute, the sub-optimal routings of some wires impacts the routes considered for others; our objective with congestion estimation is to minimize the number of poorly routed connections in the early stages of routing, and to provide an effective tie-breaking method when there are multiple routes with similar cost.

Our main contribution is the consideration of congestion estimates as part of the routing cost function. We calculate a "static" congestion estimate using methods similar to those of [10] and [14]. We also calculate a "dynamic" congestion based on the current routing solution. These estimates are combined, providing a "hint" to the global router as to which areas should be avoided (because we anticipate congestion).

The "static" and "dynamic" congestion estimates influence routing cost; prior to use, however, we *amplify* them. Motivation for this comes from the following observations. (1) In heavily congested areas, we have a strong desire to push routes to other areas. Routing in congested areas should cost substantially more than uncongested areas. (2) In lightly congested and moderately congested areas, we would wish to avoid introducing routing detours. Routing cost should not be increased in this case.

To amplify the congestion estimates, we apply a relatively simple scaling factor. For any given wire w_i in a dynamic congestion estimate, we consider the estimated demand along all routing edges used by the wire. If no routing edge has demand greater than 80% of capacity, we set the amplification factor $\alpha(w_i)$ to 0. If a routing edge has demand greater than 1.2 of capacity, $\alpha(w_i)$ is set to 1.2. Otherwise, $\alpha(w_i)$ is set to 1. The amplified dynamic demand for routing edge e_j is $d_{dynamic}(e_j) = \sum_{w_i} \alpha(w_i) \times p(w_i, e_j)$. The static amplified demand is calculated in a similar manner.

The "amplification" step increases the significance of wires in congested regions while reducing the impact of wires in non-congested regions. While the general structure of the congestion map is similar to prior methods, the congested regions are "more intense." We use this estimate to strongly influence routes in congested areas, without unduly influencing routes in less congested areas. The static estimation is performed prior to any routing, while the dynamic estimation is performed once per iteration of rip-up and reroute.

A Modified Cost Function

The static and dynamic congestion estimates are combined to influence routing cost in the following manner. We first calculate estimated congestion for routing edge e_j as described above. This estimate is then *added* to the actual routing demand used to determine route cost. We refer to this routing demand as "ambient;" it is derived directly from the congestion estimates, is persistent across an iteration of rip-up and reroute, and is not directly related to any specific wire.

During the course of multiple rip-up and reroute iterations, we then scale the ambient routing demand down. Initially, the ambient demand is "full strength," and has a substantial impact on routing; at later iterations, the effect is slight. In the routing tool by Linsker, routing cost was a linear function of routing demand. In our work, routing cost is a linear function of routing demand *plus the ambient demand, that is derived from our amplified congestion estimates.*

While the linear cost function of Linsker results in an "erosion effect," we would describe the contribution of our routing estimate as a "volcano effect." Initially, the areas where we would expect heavy congestion have routes strongly pushed away, leaving a void.

Illustration of the Approach

To illustrate the behavior of our routing tool, we present a number of "congestion maps." Lightly congested regions are shown in dark colors, while high congestion is shown in light colors. If a routing cell exceeds the target capacity, the cell is boxed in. Note that while we exceed the target capacity, that does not mean that the solution is necessarily unroutable.

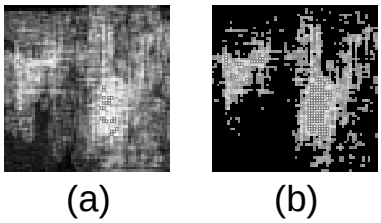

Figure 3: Original and amplified congestion estimates.

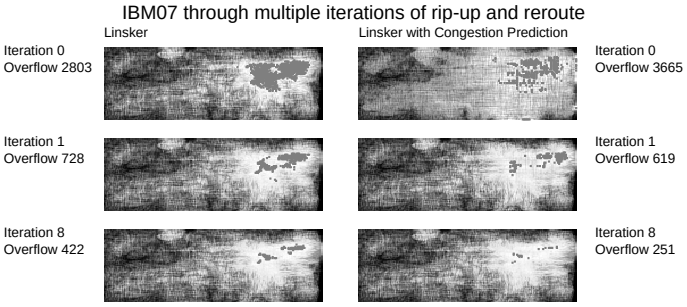

Figure 4: Congestion through repeated iterations of rip-up and reroute. The initial results obtained when using predicted congestion to influence routing cost are worse, but the routing quickly converges to significantly better solutions.

Figure 3 shows estimated congestion, and the amplified estimated congestion, for the benchmark IBM01. Figure 4 shows the congestion levels for a routing of the benchmark IBM07, at different iterations of the rip-up and reroute process. The left set of maps uses a direct implementation of Linsker-style rip-up and reroute approach. The right set shows congestion maps when we include amplified congestion estimates into the cost function.

Several observations can be made from this figure. (1) The congestion levels in the first iteration of the rip-up and reroute process are much higher for our approach. This is due to large numbers of routes detouring significantly to avoid the areas with high estimated congestion. This is the "volcano effect," where areas with the highest expected congestion may in fact have lower congestion than the surrounding regions. (2) After the first iteration, congestion levels improve dramatically, and our modified approach obtains substantially better results. (3) At termination, much of the detour introduced by the estimated congestion has been removed, resulting in only slight changes to the total wire length.

4. EXPERIMENTAL RESULTS

In this section, we present our experimental results. Of the recently published academic routers, we were able to obtain benchmarks and executable versions for only two: Labyrinth[7] and the Force-Directed router[12]. The Force-Directed router was unable to handle any of the benchmark problems correctly, so we do not report results for this tool here. We were not able to obtain the multicommodity flow based router of [1], so direct comparisons are not possible. We do not compare with commercial routing tools due to license constraints, and the fact that these tools generally optimize a number of factors other than congestion.

The Labyrinth routing tool is intended for "predictable routing,"

Benchmark	*Chi* Dispersion Router			Linsker Router			Labyrinth Predictable Router		
	overflow	wirelength	Time m:s	overflow	wirelength	Time m:s	overflow	wirelength	Time m:s
ibm01 64x64 13k nets	189	64355	00:22	249	63839	00:18	242	76228	1:12
ibm02 80x64 19k nets	66	175368	01:17	103	174610	01:08	214	202235	1:54
ibm03 80x64 26k nets	7	149695	01:03	43	149535	00:56	117	191500	2:38
ibm04 96x64 31k nets	411	170440	01:24	564	169632	01:11	786	198181	4:48
ibm06 128x64 34k nets	16	284700	02:16	55	283374	02:02	130	339379	2:51
ibm07 192x64 46k nets	251	373739	03:23	422	372451	02:52	407	450855	6:21
ibm08 192x64 49k nets	71	410507	03:06	98	409609	02:53	352	466556	6:14
ibm09 256x64 59k nets	35	420691	03:41	118	419875	03:20	310	481841	9:05
ibm10 256x64 66k nets	116	589503	05:56	211	587893	05:35	288	680113	11:19

Table 1: Comparison of our dispersion-based routing tool, a direct implementation of the approach by Linsker, and the Labyrinth predictable router, using benchmarks provided by the authors of Labyrinth.

but contains the basic elements of a traditional rip-up and reroute approach. Many routes are implemented using a simple pattern-based approach; [7] suggests that 70% of the shortest connections are appropriate for pattern routing, and this is what we use here; the remainder of the routes are found using a rip-up and reroute process with routing cost being determined by a "step" function. We have also run this routing tool without using the pattern routing option–all connections are subject to rip-up and reroute; total run times increase, but the over-congestion results do not change significantly.

To compare routing quality with other tools, we use the benchmarks provided by the authors of [7], and measure wire length and total over-congestion in the manner they suggest. These experiments are summarized in Table 1. The benchmark circuits define a rectilinear routing mesh, with fixed routing capacities on each edge. All experiments were performed on a 1.4ghz PentiumIII workstation running Linux.

Compared to Labyrinth, we obtain average wire length reductions of 15.1% and total over-congestion reductions of 65.2%. Our routing tool is roughly a factor of two faster; the original algorithm by Linsker also outperforms Labyrinth by a wide margin. We note again that these comparisons are not ideal. Unfortunately, there are relatively few global routing benchmarks, and not all published global routing tools are available for experimentation.

5. CONCLUSION

In this paper, we focus on improving congestion in global routing, using the current over-the-cell routing model. Our work improves a classic routing approach by integrating congestion estimates with the routing. By amplifying estimated congestion, we can target "problem" areas without introducing detours in uncongested regions. Compared to recent global routing work, our new approach produces excellent results. We improve over-congestion and routing length significantly. The method is also surprisingly fast, even when compared to tools that perform pattern routing. We have been extremely careful in our maze routing implementation, allowing us to consider large routing problems easily.

We have also shown that a result that was known for printed circuit board design has direct application to modern integrated circuit routing. Use of a linearly increasing cost function improves the solution quality; many current routing tools may be improved substantially by adopting this approach. Some recently published global routing tools have surprisingly poor performance when compared to this approach.

As part of our current work, we are integrating this tool into a placement and floorplanning framework.

Acknowledgements: This work was supported by the IEEC and an IBM Faculty Partnership Award. We would like to thank Ralph Linsker and Shaodi Gao of IBM, and the anonymous reviewers, for their helpful comments.

6. REFERENCES

[1] C. Albrecht. Global routing by new approximation algorithms for multicommodity flow. *IEEE Trans. on Computer-Aided Design of Integrated Circuits andSystems*, 20(5):622–631, May 2001.

[2] M. Borah, R. M. Owens, and M. J. Irwin. An edge-based heuristic for Steiner routing. *IEEE Trans. on Computer-Aided Design of Integrated Circuits andSystems*, 13(12):1563–1568, December 1994.

[3] J. Cong and P. H. Madden. Performance driven multi-layer general area routing for pcb/mcm designs. In *Proc. Design Automation Conf*, pages 356–361, 1998.

[4] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[5] Andrew Kahng and Gabriel Robins. A new class of steiner tree heuristics with good performance: the iterated 1-steiner approach. In *Proc. Design Automation Conf*, pages 428–431, 1990.

[6] Ryan Kastner, Elaheh Bozogzadeh, and Majid Sarrafzadeh. Predictable routing. *Proc. Int. Conf. on Computer Aided Design*, pages 110–113, 2000.

[7] Ryan Kastner, Elaheh Bozorgzadeh, and Majid Sarrafzadeh. An exact algorithm for coupling-free routing. In *Proc. Int. Symp. on Physical Design*, pages 10–15, 2001.

[8] C.-K. Koh and P. H. Madden. Manhattan or non-manhattan? a study of alternative vlsi routing architectures. In *Proc. Great Lakes Symposium on VLSI*, pages 47–52, 2000.

[9] J. B. Kruskal. On the shortest spanning subtree of a graph. *Proc. American Math Society*, 7:48–50, 1956.

[10] Kusnadi and Jo Dale Carothers. A method of measuring nets routability for MCM's general area routing problems. In *Proc. Int. Symp. on Physical Design*, pages 186–194, 1999.

[11] Ralph Linsker. An iterative-improvement penalty-function-driven wire routing system. *IBM Journal of Research and Development*, 28(5):613–624, September 1984.

[12] Fan Mo, Abdallah Tabbara, and Robert K. Brayton. A force-directed maze router. In *Proc. Int. Conf. on Computer Aided Design*, pages 404–408, 2001.

[13] R. C. Prim. Shortest connecting networks. *Bell System Technical Journal*, 31:1398–1401, 1957.

[14] A. Rohe and U. Brenner. An effective congestion driven placement framework. In *Proc. Int. Symp. on Physical Design*, pages 1–6, 2002.

[15] W. Swartz and C. Sechen. A new generalized row-based global router. In *Proc. Design Automation Conf*, pages 491–498, 1993.