# High-performance Routing at the Nanometer Scale

Jarrod A. Roy and Igor L. Markov
The University of Michigan, Department of EECS
2260 Hayward Ave., Ann Arbor, MI 48109-2121
{royj, imarkov}@umich.edu

**Abstract— In this work we describe significant improvements to core routing technologies and outperform the best results from the ISPD '07 Global Routing Contest, as well as previous literature, in terms of route completion, runtime and total wirelength. In particular, our router, FGR, improves upon wirelengths produced by BoxRouter and MaizeRouter in March 2007 by 9.9% and 8.4%, respectively. Additionally, we reveal the mathematical basis of negotiated-congestion routing, offer comprehensive analysis of existing routing techniques and discuss several applications at the nanometer scale.**

## 1. Introduction

Despite being one of the first areas of EDA to be automated in the 1960s, VLSI routing remains an area of active research and development as evidenced by a growing body of literature [2, 4, 10, 23, 24], recent collaboration between Cadence and IBM on routing technology [20], as well as the ISPD '07 Global Routing Contest organized by IBM Austin Research Laboratory [13]. Current efforts in routing are motivated by challenges present at the nanometer scale including: (i) very large wiring databases that require lean data structures and extremely efficient algorithms, (ii) sophisticated design rules that must be abstracted away during initial routing passes, (iii) relatively unreliable vias whose resistance may vary by up to 30 times [27], which requires via doubling [17, 18] and motivates additional effort to minimize via counts, (iv) signal integrity constraints and the dramatic impact of lateral capacitance on interconnect delay, which lead to wire density constraints, and (v) considerations of chemical mechanical polishing (CMP) that also lead to density constraints [5].

The ISPD '07 routing contest challenged the research community by distributing 16 very large routing benchmarks derived from recent chip layouts. Thanks to the wide participation in the contest and the public availability of the results, we observed an important trend — routers that achieve low wirelength often suffer high violation counts, and routers that minimize violations often produce high wirelengths. Therefore, a key focus of our work is on adequate pricing of routing resources to balance interconnect length and congestion in multi-million gate designs, in a way that also allows to trade-off other nanoscale objectives and constraints. Additionally, the effective handling of vias, multiple metal layers and other aspects of nanoscale routing pose a series of algorithmic, implementation, benchmarking and integration challenges.

In this work we develop a high-performance routing technique based on Discrete Lagrange Multipliers (DLM), while pointing out inaccuracies, limitations and pitfalls of the related technique known as negotiated-congestion routing [21]. In particular, DLM offers a natural way to handle net weights and timing optimization in routing, and explains several empirical effects observed in negotiated-congestion techniques such as the last-gasp problem and the relative simplicity of 2-d formulations compared to multi-layer (3-d) formulations. Proposed algorithms are implemented in FGR[1], a high-performance global router for nanometer scale designs.

Our key contributions are:

- A routing technique based on Discrete Lagrange Multipliers (DLM) which provides a natural way to handle net weights and timing optimization in global routing. FGR handles two- and three-dimensional routing of ASICs with up to 870,000 nets.[2]
- Extensions of A*-search to restructure net topologies so as to avoid congestion and circumvent obstacles.
- Improved wirelength on the ISPD '07 Global Routing Contest suite [13]. FGR produces smaller wirelengths than the winners of the contest *on every benchmark*, and is able to route without overflows every benchmark that the winners routed without overflows. In terms of wirelength, FGR outperforms BoxRouter [4] by 9.9% and MaizeRouter [22] by 8.4%.
- Violation-free routing of all ISPD '98 IBM benchmarks, unlike routers in previous literature. FGR uses 35% less runtime than BoxRouter and produces solutions with 2.7% smaller wirelength.
- Accurate congestion estimation which is extremely important at the nanometer scale.
- Thorough empirical evaluation of several routing strategies and algorithms including net decomposition by MST vs. Steiner trees and layer assignment for 3-d routing problems vs. direct 3-d maze routing. We identify previously unreported bottlenecks, such as the "last gasp" problem in negotiated-congestion routing, and propose solutions.

This paper is organized as follows. In Section 2, we review relevant background and previous work. Next in Section 3 we describe the architecture of the FGR router, the mathematical basis for its key algorithms, and important insights into the integration of major components. In Section 4 we outline applications of FGR to several areas of physical design. We benchmark FGR against state of the art in Section 5 and conclude in Section 6.

## 2. Background and Previous Work

Routing plays a key role in VLSI physical design as it determines the specific shape and layout of interconnect, impacting performance, power and manufacturability. Routing is traditionally divided into the two steps of global and detail routing.

**Global and Detail Routing.** During global routing, complex design rules are abstracted away and a design is divided into a regular grid (see Figure 1). Routes are created for each net that connect adjacent grid cells. Capacities are assigned to pairs of adjacent grid cells to model limited routing resources between the cells. Since different metal layers may use distinct wire pitches, routing capacities at each layer may differ to reflect this. A global routing solution is legal if all nets are connected and all capacity constraints are satisfied.

Detail routing takes a global routing solution with a small number of capacity violations (overflows), or none at all, and assigns wires to routing tracks while enforcing spacing constraints and more sophisticated design rules. Starting with slightly illegal global routes can make detail routing considerably more difficult, therefore a global router must minimize violations and wirelength, seeking to avoid violations entirely when possible.

Traditional algorithms for detail routing often assume a specific, small number of metal layers and operate in isolated layout regions — channels or switch-boxes. However, over-the-cell routing with six or more metal layers made many such algorithms obsolete and lead to the adoption of similar graph-theoretical techniques in global and detail routing, perhaps with different layout, resource and delay models.

[1] "Fairly Good Router"

[2] This is almost an order of magnitude greater than what has been reported in the literature for most ASIC and FPGA routers. In the 32-bit address space, FGR scales up to 1,000,000 nets.

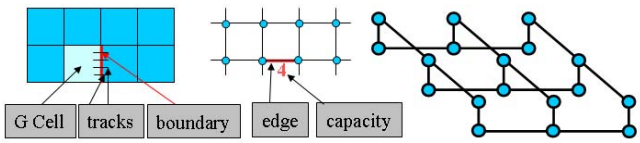



Figure 1. Pictorial representations of the global routing grid. The images on the left and in the middle show how the layout is abstracted into a regular grid of GCells. GCells are represented by vertices, with adjacent vertices connected by graph edges. Capacities on edges that join GCells can be defined as the number of routing tracks that cross GCell boundaries. The image at the right shows that horizontal and vertical connections can be on different layers with vias connecting the layers.

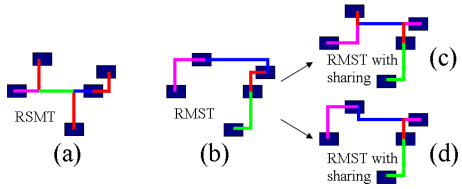



Figure 2. Decomposition of a 5-pin net by minimal Steiner tree (a), MST (b) and MSTs with sharing (c)&(d). The choice of (c) or (d) depends on congestion. The minimal Steiner tree (a) contains 5 flat subnets and 1 L-shaped subnet, whereas the shared MST (d) has 2 flat subnets and 3 L-shaped subnets which gives it greater flexibility.

In our experience with Cadence WarpRoute, three quarters of total runtime are spent in detail routing, but the quality of global routes profoundly affects the runtime and success of detail routing. A recent proposal [23] suggests invoking a fast global router during global and detail placement, so as to mitigate wiring congestion early. This application is particularly attractive for sub-130nm technology nodes where lateral capacitance of wires is a major contributor to interconnect delay. In this context, accurate timing analysis requires information about regions through which a given net passes as well as wire density in these regions [30].

**Maze Routing** connects pairs of terminals on the routing grid using standard search techniques such BFS and Dijkstra's algorithm [7]. More than 50% of nets in modern designs connect only two pins. BFS can find the shortest path between a source location and a target location, if one exists, but cannot handle routing segments with non-trivial weights. Dijkstra's algorithm can handle non-negative routing segment costs, but is at least several times slower than BFS. A*-search is a modification to Dijkstra's algorithm that significantly improves speed during 2-d and 3-d routing [11]. In A*-search, a lower bound of the distance to the target is added to node priority in Dijkstra's algorithm. Straight-line distance is commonly used as a lower bound.

**Pattern Routing** [15] is a technique that severely restricts the number of ways in which a net can be routed to simplify the routing process. For example, L-shape routing seeks to implement each two-pin net with a single bend. This technique is surprisingly useful in ASIC routing and justified by via minimization. Empirical studies [32] show that in a fully-routed design a majority of all 2-pin nets take on L-shapes. In global routing, where minor detours are abstracted away, L-shapes are even more prevalent. Two-bend routes are often called Z-shapes, but generic pattern-based routing can consider any finite number of routing topologies for each net, and selects one of them. It is particularly amenable to Integer Linear Programming formulations [4], as described later in the section.

**Multi-pin nets.** Most global routing algorithms decompose nets with three or more pins into two-pin subnets at the beginning of global routing as this eases maze routing. This decomposition has been traditionally done using Minimal Spanning Tree (MST) algorithms, but using fast and extremely accurate Rectilinear Steiner Minimal Tree (RSMT) construction algorithms has become increasingly popular in the literature [4, 23, 24]. Four decompositions for a 5-pin net based on Steiner trees and MST are shown in Figure 2.

The RSMT tool FLUTE [6] is used in both BoxRouter [4] and FastRoute [23, 24]. FLUTE uses look-up tables for nets with nine or fewer pins and quickly builds optimal trees for such nets [6]. For larger nets, a divide-and-conquer method is employed [6]. FastSteiner [14] is another RSMT algorithm that is more scalable than most RSMT algorithms. FastSteiner does not guarantee optimality, but frequently produces solutions with smaller total wirelength than FLUTE for nets with more than nine pins. In Section 5.3 we compare MST with sharing to a combination of FLUTE and FastSteiner and find that Steiner constructors lead to smaller routed length but greater via counts.

**Rip-up-and-re-route (RRR)** takes an initial, usually illegal, routing solution and iterates greedy one-net-at-a-time routing passes for nets that compete for routing resources, but may change the ordering each time in hope to better reconcile these nets. In each iteration, nets that pass through congested regions are "ripped up" (all resources for the net are removed from the routing grid) and are rerouted with a maze router to use lesser congested regions. Major differences between various implementations [4, 8, 10, 21, 23, 24] include which nets are ripped up and rerouted at each iteration, the order in which to rip up nets and reroute them, if nets are allowed to be rerouted through areas that are already congested, and the costs associated with routing through a particular routing edge given its current congestion.

**Congestion Amplification** [10] was recently introduced as an improvement to pricing of routing resources during RRR. Many routers that employ RRR do not penalize nets for going through uncongested regions, and then drastically increase cost once a routing edge is full. The authors of [10] propose to use a more gradual linear cost function for edges before they become full in order to spread wires from areas that are likely to become congested. In addition, when congestion estimates are calculated after each iteration of RRR, regions with high congestion have their estimates artificially increased (amplified) and regions with low congestion have their estimates decreased. This provides a greater incentive for maze routers to avoid highly congested regions, often at the cost of increased wirelength.

**Negotiated-congestion Routing (NCR)** [21] was introduced in the mid-1990s for global routing in FPGAs, but has not seen much use in the ASIC literature. NCR builds upon RRR by gradually making routing edges that are consistently congested more expensive, encouraging the maze router to choose alternative routes when they are available. The cost $c_e$ of routing edge $e$

$$c_e = (b_e + h_e) \cdot p_e \tag{1}$$

is a function of the intrinsic cost ($b_e$), added cost reflecting congestion history ($h_e$), and penalty for current congestion ($p_e$) [21]. NCR seeks to minimize $\sum_e c_e$.

To begin negotiated-congestion routing, each net is routed using the smallest possible wirelength regardless of edge capacities. Next, rip-up-and-re-route proceeds. At the beginning of a RRR iteration, the historical cost $h_e$ of all over-capacity routing edges is increased:

$$h_e^{k+1} = \begin{cases} h_e^k + h_{inc} & \text{if } e \text{ has overflow} \\ h_e^k & \text{otherwise} \end{cases} \tag{2}$$

where $h_{inc}$ is a constant. Each net of the design is then individually ripped up and rerouted by a maze router. The authors suggest that only nets passing through congested regions need to be rerouted and we take this approach in FGR. The ordering of nets during rip-up-and-re-route is the same for each iteration, but can be chosen arbitrarily, according to the authors of [21], because the gradual cost increase in congested areas removes ties that require sophisticated net ordering techniques in traditional RRR implementations.

**FastRoute** [23, 24] uses a simplified, more greedy form of RRR and finishes orders of magnitude faster than other routers. However, it was able to legally route only 6 of 16 benchmarks at the ISPD '07 contest [13], while other routers completed up to 12 benchmarks without violations. Additionally, on the easier ISPD '98 benchmarks, it routes fewer benchmarks than FGR (see Table 1).

FastRoute 1.0 [23] first uses FLUTE to decompose nets and estimate congestion in the design, then attempts to restructure Steiner

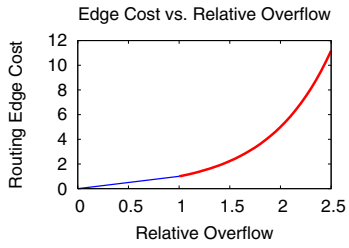



Figure 3. Cost of a routing edge as a function of relative overflow. Cost is linear while the edge is not overfilled, but grows exponentially once the edge has overflow.

trees to avoid congestion. FastRoute 2.0 [24] features the following modification of RRR. When a single subnet is ripped up, the net to which the subnet belongs will be separated into two connected components. It becomes the maze router's job to connect the two components of the net in the least costly way. While this optimization allows the router to move Steiner points away from congested regions, it invalidates the point-to-point lower bound on which A*-search relies. Therefore, the slower Dijkstra's algorithm must be used instead.

**BoxRouter** [4] avoids fine-grain net ordering in congested regions through the use integer linear programming (ILP) formulations. BoxRouter decomposes nets using Steiner trees produced with FLUTE but never re-examines their decomposition. Next it performs a pass of pattern routing that identifies the most congested rectangular region, where it formulates an ILP to route as many nets using L-shapes as possible. Remaining nets are routed by the maze router, using as few resources outside the region as possible. Next, the region is expanded, and an incremental ILP formulation is used. This cycle repeats until the entire layout is covered by the expanding region. Traditionally ILP solving is considered slow, and ILPs have difficulty expressing non-linear delay models.

**Other techniques** for routing have been proposed, such as the use of multi-commodity flows (MCF) [2, 11]. ILP-based BoxRouter has been compared to a recent MCF-based router [4] and found to be superior in speed and solution quality. Additionally, MCF techniques offer less flexibility in terms of objective functions and constraints than the RRR and NCR frameworks.

## 3. High-performance Global Routing

In this section we describe the architecture of FGR, the mathematical basis for its key algorithms, and notable implementation insights.

### 3.1 Basic Algorithmic Framework

Routing algorithms must carefully balance wirelength minimization and congestion. Some detours may be necessary to avoid routing violations and overcapacity GCells, but excessive detouring leads to overconsumption of routing resources, aggravating congestion. In particular, the results of the ISPD '07 routing contest [13] show that some routers are good at finding violation-free solutions, some are good at minimizing wirelength, but few are good at both. A likely source of this inflexibility is the common use of uniform, predetermined rules in all regions of the chip as in FastRoute [23, 24] and the Chi dispersion router [10].

In continuous optimization, dynamic pricing of constraint satisfaction can be modeled by Lagrange multipliers — a mathematical method for optimizing a multivariate function subject to a number of constraints [16]:

$$\begin{array}{c} \min_{\mathbf{x}\in X} W(\mathbf{x}) \\ \text{subject to } C_e(\mathbf{x}) = 0, \quad 1 \le e \le n \end{array} \tag{3}$$

The constrained optimization is reduced to the unconstrained optimization of the Lagrangian function $F$

$$F(\mathbf{x},\lambda) = W(\mathbf{x}) + \sum_{e=1}^{n} \lambda_e C_e(\mathbf{x}) \tag{4}$$

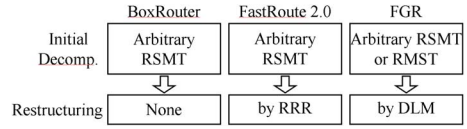



Figure 4. Net decomposition techniques used by BoxRouter [4], FastRoute 2.0 [24] and FGR. In Section 5.3, we compare the use of RMSTs and RSMT in FGR.

where $\lambda = (\lambda_1, \ldots, \lambda_n)$ are real-valued Lagrange multipliers. In the case of routing, $C_e(\mathbf{x})$ represents the overflow of routing edge $e$. $W(\mathbf{x})$ represents the total wirelength of routing solution $\mathbf{x}$ and is usually defined as a sum over nets or routing edges

$$W(\mathbf{x}) = \sum_{i=1}^{m} R_i(\mathbf{x}) = \sum_{e=1}^{n} B_e(\mathbf{x}) \tag{5}$$

where $R_i(\mathbf{x})$ is the number of segments used by net $i$ and $B_e(\mathbf{x})$ is the number of nets passing through edge $e$. Thus (4) can be rewritten

$$F(\mathbf{x},\lambda) = \sum_{e=1}^{n} (B_e(\mathbf{x}) + \lambda_e C_e(\mathbf{x})) \tag{6}$$

Here both original unknowns $\mathbf{x}$ and the Lagrange multipliers $\{\lambda_e\}$ are considered variables subject to optimization. For large sparse convex problems iterative techniques are used, such as *steepest descent*, *Newton's method*, etc. In particular, Lagrange multipliers are updated additively as follows

$$\lambda^{k+1} = \lambda^k + \alpha C(\mathbf{x}^k) \tag{7}$$

where $\alpha > 0$ is a line-search parameter. Note the similarity in the update of the Lagrange multipliers and how $h_e$ is updated in Formula 2. While we are also dealing with large sparse problems, they are discrete and non-convex. This calls for a different iterative optimization procedure, such as *greedy search*, *hill-climbing* or *rip-up-and-reroute*. However, since Lagrange multipliers remain continuous, the same update rule can be adopted.[3]

Interpreting Formula 6 for a given net $i$ in terms of NCR yields

$$c_e = b_e + h_e \cdot p_e \tag{8}$$

which is different than Formula 1 [21], but also makes more sense since it preserves the base cost. Therefore FGR uses this Discrete Lagrange Multiplier (DLM) formulation instead of NCR which was used in FGR's ISPD '07 contest submission. To compute $p_e$, we use a new penalty function introduced in Section 2 below. Furthermore, the justification of dynamic cost updates through DLMs explains the results we see in Sections 3.4, 3.5 and 5.

### 3.2 Congestion Penalty

Let $r_e$ and $u_e$ represent the resources and current usage of a routing edge $e$ and define the relative overflow $\omega_e = u_e/r_e$. We compute the congestion penalty term $p_e$ for edge $e$ as a function of $\omega_e$.

$$p_e = \begin{cases} \exp\left(k(\omega_e - 1)\right) & \text{if } \omega_e > 1 \\ \omega_e & \text{otherwise} \end{cases} \tag{9}$$

The exponential nature of our cost function for routing edges with overflow serves to amplify congestion and gives the maze router incentive to avoid overfull edges when re-routing nets (see Figure 3, where $k = \ln 5$). We have studied $0 < k \le \ln 10$ and found that higher values of $k$ reduce runtime, but increase detouring and routed length. FGR uses $k = \ln 5$ by default. Instead of routing all nets by their shortest paths to find an initial routing solution, which is common in NCR, FGR uses $b_e + p_e$ as the weight for edges to create an initial solution.

[3] To this end, the use of Lagrange multipliers can be viewed as a way to leverage continuous optimization in a discrete domain, such as nanoscale routing.

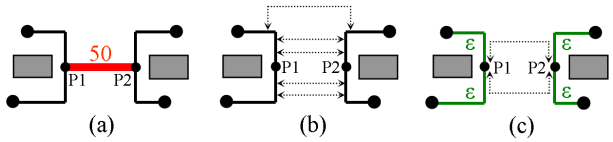



Figure 5. Re-routing a subnet and changing net topology in FGR. The shaded boxes represent obstacles. The tree in (a) passes through a congested segment in the middle which must be ripped up. The dashed arrows in (b) represent several possible re-routings that a restructuring algorithm may consider. The re-routings shown in (c) are two that FGR will consider during DLM. Paths considered by FGR must start and end along the endpoints of the segment that was removed. Both of these re-routings reuse routing segments from the net and create new Steiner points if chosen. The use of temporary zero-cost edges is required to preserve the efficiency of A*-search.

```
Input:  2-d routing solution, 2dsol
Output:  3-d routing solution, 3dsol
1  foreach net n in 2dsol
2    foreach subnet s of n
3      route = 2dsol.getRoute(s)
4      currPoint = s.terminal1
5      currLayer = currPoint.layer
6      while(currPoint != s.terminal2)
7        nextPoint = route.getNextPoint(currPoint)
8        find nextLayer:  the layer closest to
           currLayer where adding an edge connecting
           currPoint and nextPoint causes least overflow
9        add segment from currPoint to nextPoint on layer nextLayer to 3dsol
10       add vias connecting (currPoint.x,currPoint.y,currLayer)
           and (currPoint.x,currPoint.y,nextLayer) to 3dsol
11       currPoint = nextPoint
12       currLayer = nextLayer
13     add vias connecting (currPoint.x,currPoint.y,currLayer)
         and (currPoint.x,currPoint.y,s.terminal2.layer) to 3dsol
```

Figure 6. Layer assignment in FGR.

### 3.3 Interactions Between Single- and Multi-Net Routing

FGR initially decomposes nets using an RSMT or RMST topology. However, given that congestion-driven Steiner trees are not easy to construct and precise congestion in every GCell is not known beforehand, we found it important to modify net topology during routing.

Figure 4 compares the net decomposition and restructuring techniques used by FGR to those in prior work. During DLM, the most congested subnets are ripped up and rerouted by A*-search. When ripping up a subnet with endpoints P1 and P2, FastRoute 2.0 tries to reconnect the two components of the net, not necessarily using P1 or P2, which invalidates the point-to-point lower bound used in A*-search. When re-routing a subnet, FGR requires the replacement segments to pass between P1 and P2, but we modify the A*-search as follows. We add a single check when pricing routing edges: if the routing edge we are considering is already in use by a sibling of the current subnet (sibling subnets belong to the same net), then we temporarily set its cost to zero. This allows the maze router to re-connect the net through any pair of points connected to P1 and P2.

Temporary change of edge costs to 0 is easy to implement during A*-search because we route one net at a time and can undo any cost adjustments before considering other nets. However, in order to use A*-search, we must supply a correct lower bound. We normally use the 3-d Manhattan distance multiplied by the minimum cost of any routing segment. The naive solution — to ignore the 0-cost edges — may produce estimates that are greater than the true cost, which would invalidate A*-search. However, if we literally set an edge's cost to zero, the lower bound will automatically become zero. Therefore, in our implementation we set the cost of previously used edges to $\varepsilon > 0$, a very small value. This technique is illustrated in Figure 5, where FGR modifies the net topology to avoid congestion.

While prior state-of-the-art routers (BoxRouter, FastRoute and MaizeRouter) consistently start by decomposing multi-pin nets with minimal Steiner trees, we believe that our integration of topology restructuring into a powerful DLM framework facilitates additional opportunities. As illustrated in Figure 2, Steiner trees tend to generate net decompositions with many flat subnets which offer no flexibility in routing. MSTs tend to have fewer edges but with more flexibility, which can be exploited by DLM to avoid congestion. Moreover, the gradual addition of sharing to MSTs during DLM-based topology restructuring can generate high-quality congestion-driven Steiner trees without the need to estimate congestion before routing. Starting with minimal Steiner trees seems to require heavier restructuring to achieve similar effects, and could not only slow down maze routing, but also make RRR or DLM less successful. The use of RSMTs and RMSTs is compared in Section 5.3.

### 3.4 Overcoming the "Last Gasp" Problem

Discrete Lagrange multipliers work well at the large scale because the statistical behavior of numerous discrete variables is not very different from the continuous case. However, when only several violations remain, the routing task becomes much more discrete.

In our experiments with almost every benchmark, we have observed unusual behavior where FGR spends many DLM iterations with a nearly legal solution before it is able to terminate with a completely legal solution. Indeed, more than 75% of DLM's iterations for the adaptec2 benchmark [13] take place when less than 0.01% of routing segments have overflow. We term this undesirable behavior the "last gasp" problem.

To rectify this situation, we propose the following improvement to negotiated-congestion routing. When the percentage of routing edges with overflow becomes small, we restrict the maze router to using only edges that have available space and weigh routing edges only by their base cost $b_e$. Thus if there is any way to route the net without causing overflow, we will take it, to avoid further rip-up iterations. If no path is possible, default DLM is used. This last phase of DLM reduces iterations without noticeably impacting total routed wirelength.

### 3.5 Three-dimensional Routing

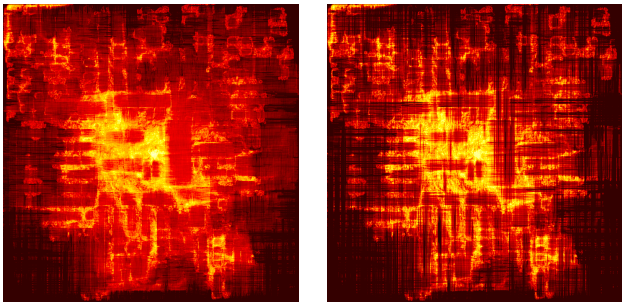
The difficulties experienced by DLM due to discreteness also suggest that traditional 2-d routing may be considerably easier than proper 3-d routing where smaller edge capacities are spread through multiple routing layers. In other words, aggregating edge capacities in one layer would encourage continuous-like resource pricing, making it easier to satisfy all constraints. This is consistent with what we observe in experiments discussed in Section 5.4.

FGR performs 3-d routing by first projecting the routing instance onto a 2-d grid and aggregating the capacities of edges that project onto each other. This grid contains a single layer of horizontal wires and a single layer of vertical wires connected by a layer of vias, such as grid depicted at the right of Figure 1. Capacities on higher layers may be smaller due to increased pitch, but for each routing grid edge, we calculate the number of wires that are allowed to pass through it when aggregating, which takes wire widths and pitches into account. FGR routes this 2-d problem instance as normal until a legal solution is found or a runtime/iteration limit is reached. Next FGR performs layer assignment for each routing segment used in the 2-d solution.

**Theorem 1** *If the projected 2-d instance has a legal solution and vias are unconstrained, the original 3-d instance has a legal solution.*

**Proof:** 3-d routes can be constructed by the algorithm in Figure 6.

FGR's method will produce a 3-d solution that uses exactly the same number of routing segments as the 2-d solution, but differ in via counts. Unfortunately the difference in via counts is usually large and proportional to the number of layers in the 3-d instance. As such, we perform a single round of RRR for every subnet to reduce vias. In this round of optimization, the cost of each routing segment is much simpler than in DLM: each routing segment is assigned a cost of 1 and vias are assigned an appropriate relative cost (for example, one via costs as much as three routing segments in the ISPD '07 Contest, so vias would have a cost of 3). It is easy to lower-bound the cost of a path with these edge costs, so it is particularly amenable to A*-search (the lower bound is the 3-d Manhattan distance). Each subnet is ripped up and rerouted with the maze router individually and edges with no spare capacity are not allowed.



(a) (b)

Figure 7. Congestion on the adaptec4 2-d benchmark [13] (a) by FGR in estimation mode and (b) after full violation-free routing by FGR.

| Bench-mark | BoxRouter [4] ovfl | BoxRouter [4] WL | BoxRouter [4] time (s) | FastRoute 2.0 [24] ovfl | FastRoute 2.0 [24] WL | FGR ovfl | FGR WL | FGR time (s) |
|---|---|---|---|---|---|---|---|---|
| ibm01 | **102** | 65588 | 6 | **31** | 68489 | 0 | **63332** | 10 |
| ibm02 | **33** | 178759 | 25 | 0 | 178868 | 0 | **168918** | 13 |
| ibm03 | 0 | 151299 | 13 | 0 | 150393 | 0 | **146412** | 5 |
| ibm04 | **309** | 173289 | 18 | **64** | 175037 | 0 | **167101** | 29 |
| ibm05 | 0 | 409747 | 37 | – | – | 0 | **409739** | 6 |
| ibm06 | 0 | 282325 | 25 | 0 | 284935 | 0 | **277608** | 18 |
| ibm07 | **53** | 378876 | 39 | 0 | 375185 | 0 | **366180** | 20 |
| ibm08 | 0 | 415025 | 68 | 0 | 411703 | 0 | **404714** | 18 |
| ibm09 | 0 | 418615 | 50 | **3** | 424949 | 0 | **413053** | 20 |
| ibm10 | 0 | 593186 | 73 | 0 | 595622 | 0 | **578795** | 92 |
| Average | | **+2.71%** | | | **+3.64%** | | | |

Table 1. Comparison of FGR to FastRoute 2.0 and BoxRouter on the ISPD '98 IBM benchmark suite [12]. FGR completes all 10 of the benchmarks while BoxRouter [4] and FastRoute 2.0 [24] leave overflow on 4 and 3 of the benchmarks, respectively. In terms of routed wirelength, FGR outperforms BoxRouter by 2.7% and FastRoute 2.0 by 3.6%. Runtimes for BoxRouter and FGR are given in seconds. FastRoute runtimes are not listed as binaries are unavailable. FGR is faster than BoxRouter on 7 of the 10 benchmarks and uses 35% less runtime.

## 4. Applications

We now discuss applications of proposed techniques for high-performance routing to congestion and delay estimation, timing-driven routing, as well as detail routing of ASICs and FPGAs.

**Congestion and Delay Estimation.** Lateral capacitance of wires is a major contributor to interconnect delay in sub-130nm technology. In this context, accurate timing analysis can use information about regions through which a given net passes as well as wire density in these regions [30]. Accurate wire density information is especially important to properly account for crosstalk [25].

Recent work on probabilistic congestion estimation used the assumption that L- and Z-shaped routes were the most common routes taken by industrial routers [32]. Unfortunately, these routing statistics differ significantly from what we observed in detailed layouts of realistic circuits placed with Cadence QPlace and routed with Cadence WarpRoute / NanoRoute. In fact, more recent work by Westra and Groeneveld has questioned the usefulness of probabilistic congestion estimators [33], including the technique presented in [32].[4] They find poor correlation between probabilistic estimates and final routing congestion for both routable and especially unroutable designs [33]. To rectify the situation, the authors propose to use a fast global router as a congestion estimator — the idea more recently proposed in [23]. FGR can also be used to create accurate congestion estimates. Since FGR uses essentially the same edge costs during initial routing as during DLM and reroutes congested areas gradually, we find that congestion directly after the initial routing phase correlates very well with final routing congestion (see Figure 7) The initial routing phase is at least 10 times faster than entire global routing, but its results can accurately predict congestion in final routing solutions making it valuable during upstream stages of physical design and physical synthesis. In its estimation mode, FGR also performs layer assignment which is crucial for accurate delay estimation at the nanometer scale due to varying wire thickness, buffer requirements, and higher usage of vias (which have high resistance, defect rates and variability).

In addition to delay estimation, FGR can also be used to reduce delay in a circuit. In particular, to avoid detouring critical nets, FGR can route them preferentially as follows. Firstly, FGR can order these nets earlier during the initial routing phase. The majority of routing resources will be unused early in the initial routing phase, so the critical nets will have a greater likelihood of routing without detours. After the initial routing phase, FGR can require that these nets only be routed within their bounding boxes. Delay budgets per net can be modeled as constraints (with associated Lagrange multipliers) and nets which do not meet their budgets can be ripped up at each DLM iteration and rerouted.

**Handling Density Constraints.** Given that wire density impacts lateral capacitance [30] and crosstalk [25], it is natural to consider density constraints as well as resource constraints during global routing. Edge capacities can be reduced to combat high wire densities, but this can make routing unnecessarily difficult. FGR's techniques are sufficiently general to directly handle a wide range of constraints, including density constraints. An additional cost (Lagrange multiplier) can be added per routing edge which represents a penalty for violating density constraints, similar to the historical costs for congestion. At every iteration where a routing edge is part of an overly dense region, these new costs can be gradually increased. As this modified form of DLM progresses, dense regions will gradually become more expensive, causing the maze router to avoid them.

**Detail Routing for ASICs and FPGAs.** Traditional algorithms for channel routing have become less relevant in the context of over-the-cell routing, and even less relevant given the complicated design rules imposed by manufacturability considerations. In contrast, the algorithmic paradigms in FGR naturally extend to over-the-cell detail routing on a grid of tracks. In particular, A*-search can handle spacing constraints found in common design rules: routing segments that are too close to used tracks can have their costs significantly increased or can be temporarily blocked in the routing graph.

We note that negotiated-congestion routing (NCR) was originally developed for global routing of FPGAs [21], but in most recent work [29] has been extended with A*-search and applied to hard instances of detail routing for FPGAs. All analysis and improvements to NCR reported in our work are directly applicable in that context. Moreover, the constraint-driven nature of DLM provides a generic way to handle new constraints, including those endemic to FPGAs and ASICs.

## 5. Experimental Results

We have implemented FGR in C++ without external libraries (compiled with GCC 3.4.5), but added optional interface to the Steiner-tree packages FLUTE [6] and FastSteiner [14] to compare them with MST decompositions. The core algorithms and data structures of FGR were implemented in one month. All runs were performed on 2.4 GHz Opteron processors with at least 4GB of RAM.

### *5.1 Performance on ISPD '98 Benchmarks*

Table 1 compares FGR to BoxRouter and FastRoute 2.0 [24] on the ISPD '98 benchmarks [12]. Unlike all previous routers in the literature, FGR routes all of the IBM designs without overflow. Both BoxRouter and FastRoute 2.0, which report the best results on this suite so far, produce solutions with overflow on 4 and 3 of the benchmarks, respectively. Overall, FGR produces solutions with 2.72% shorter wirelength than BoxRouter and 3.62% shorter than FastRoute 2.0. FGR is faster than BoxRouter on 7 of the 10 benchmarks and uses 35% less runtime to complete the entire suite. Unlike the ISPD '07 contest benchmarks, the ISPD '98 benchmarks feature only a single metal layer, making via minimization unnecessary.

### *5.2 Performance on ISPD '07 Benchmarks and Impact of Vias*

The ISPD '07 Global Routing Contest benchmarks [13] are considerably larger than the ISPD '98 benchmarks and include both two- and three-dimensional variants. These benchmarks feature non-trivial

[4] An additional deterrent to using probabilistic congestion estimation in the industry is that it has been patented in 1994/96 [31].

| Bench-mark | | Best of BoxRouter and MaizeRouter Overflow total | Overflow max | Cost (e5) | Router | FGR Overflow total | Overflow max | Cost (e5) | vs. Best |
|---|---|---|---|---|---|---|---|---|---|
| adaptec | #1 2-d | 0 | 0 | 58.84 | Box | 0 | 0 | **53.81** | -8.55% |
| | #1 3-d | 0 | 0 | 99.61 | Maize | 0 | 0 | **88.39** | -11.26% |
| | #2 2-d | 0 | 0 | 55.69 | Box | 0 | 0 | **51.86** | -6.88% |
| | #2 3-d | 0 | 0 | 98.12 | Maize | 0 | 0 | **89.89** | -8.39% |
| | #3 2-d | 0 | 0 | 137.75 | Maize | 0 | 0 | **129.58** | -5.93% |
| | #3 3-d | 0 | 0 | 214.08 | Maize | 0 | 0 | **199.60** | -6.76% |
| | #4 2-d | 0 | 0 | 128.45 | Maize | 0 | 0 | **124.12** | -3.37% |
| | #4 3-d | 0 | 0 | 194.38 | Maize | 0 | 0 | **179.36*** | -7.73% |
| | #5 2-d | 0 | 0 | 164.32 | Box | 0 | 0 | **150.64** | -8.33% |
| | #5 3-d | 0 | 0 | 298.08 | Box | 0 | 0 | **259.89** | -12.81% |
| newblue | #1 2-d | 400 | 2 | 51.13 | Box | 452 | 4 | **47.43** | -7.24% |
| | #1 3-d | 400 | 2 | 101.83 | Box | 452 | 2 | **94.27** | -7.42% |
| | #2 2-d | 0 | 0 | 79.64 | Maize | 0 | 0 | **75.87** | -4.73% |
| | #2 3-d | 0 | 0 | 139.66 | Maize | 0 | 0 | **129.40*** | -7.35% |
| | #3 2-d | 32588 | 1236 | 114.63 | Maize | 38580 | 1120 | **109.34** | -4.61% |
| | #3 3-d | 32840 | 1058 | 184.40 | Maize | 38580 | 374 | **173.82** | -5.74% |
| Average | | | | | | | | | **-7.35%** |

Table 2. Comparison of FGR to the other top-3 routers at the ISPD '07 Global Routing Contest [13]. FGR routes as many benchmarks without overflow as the winners of the contest with 7.4% better wirelength than the best of BoxRouter [4] and MaizeRouter [22]. *These benchmarks were routed using FGR's option "-full3d".

| Benchmark | Decomposition by MST Segment WL (e5) | Vias (e5) | Total cost | Time (m) | Decomposition by Steiner trees Segment WL (e5) | Vias (e5) | Total cost | Time (m) |
|---|---|---|---|---|---|---|---|---|
| adaptec1 2-d | 35.88 | 6.19 | 54.44 | 451 | 35.78 | 6.24 | 54.49 | 403 |
| adaptec1 3-d | 36.37 | 17.36 | 88.45 | 430 | 36.26 | 18.04 | 90.37 | 395 |
| adaptec2 2-d | 33.21 | 6.36 | 52.30 | 56 | 33.10 | 6.43 | 52.38 | 170 |
| adaptec2 3-d | 33.74 | 18.72 | 89.89 | 64 | 33.62 | 19.37 | 91.72 | 168 |
| adaptec3 2-d | 96.09 | 11.60 | 130.89 | 179 | 95.55 | 11.67 | 130.57 | 222 |
| adaptec3 3-d | 97.02 | 34.21 | 199.66 | 243 | 96.42 | 35.49 | 202.90 | 281 |
| adaptec4 2-d | 90.02 | 11.66 | 125.00 | 19 | 89.37 | 11.72 | 124.53 | 18 |
| adaptec4 3-d | 91.28 | 30.56 | 182.96 | 55 | 90.59 | 31.59 | 185.35 | 58 |
| adaptec5 2-d | 102.79 | 16.45 | 152.13 | 713 | 102.56 | 16.63 | 152.45 | 771 |
| adaptec5 3-d | 103.89 | 52.03 | 259.98 | 740 | 103.62 | 53.78 | 264.97 | 796 |
| newblue1 2-d | 24.15 | 7.76 | 47.42 | 1441 | 24.00 | 7.74 | 47.22 | 1441 |
| newblue1 3-d | 24.15 | 23.37 | 94.26 | 1442 | 24.00 | 24.00 | 96.01 | 1442 |
| newblue2 2-d | 46.81 | 9.90 | 76.51 | 4 | 46.41 | 9.95 | 76.27 | 4 |
| newblue2 3-d | 47.91 | 28.08 | 132.16 | 10 | 47.51 | 29.08 | 134.75 | 10 |
| newblue3 2-d | 75.63 | 11.20 | 109.23 | 1555 | 75.24 | 11.15 | 108.71 | 1460 |
| newblue3 3-d | 75.63 | 32.69 | 173.71 | 1501 | 75.24 | 33.04 | 174.35 | 1462 |
| Ratio | | | | | -0.52% | +1.81% | +0.74% | +22.0% |

Table 3. Comparing net decomposition by MST vs. Steiner trees on the ISPD '07 benchmarks [13]. Time taken for decomposition by MST or Steiner trees is less than 1 minute on all benchmarks and does not impact runtimes.

routing obstacles, and, consequently, routing resources are not spread evenly as in the ISPD '98 suite. Table 3 shows runtimes for FGR on these benchmarks. In all cases FGR stays within the 32-bit memory space and finishes well under a given 24-hour timeout on all but the newblue1 and newblue3 benchmarks for which no legal solutions were found at the ISPD '07 contest.[5]

We compare FGR to the routers that scored best at the ISPD 2007 contest and results are shown in Table 2. Since an earlier version of FGR placed 1st in the 2-d and 3rd in the 3-d categories, we exclude it from comparison (however, the version we report improves upon FGR's contest results on every benchmark). MaizeRouter [22] placed 1st in 3-d and 2nd in 2-d, and BoxRouter placed 2nd in 3-d and 3rd in 2-d at the contest. FGR produces smallest wirelengths *on every benchmark* and is able to route without overflow every benchmark that was legally routed at the contest. In particular, FGR outperforms BoxRouter in wirelength by 9.9% and MaizeRouter by 8.4%.

Via counts are included in the total cost of a routing solution to both the 2-d and 3-d variants of the contest benchmarks such that one via costs the same as three routing segments. Vias represent from 26% to 49% of the total cost of FGR's solutions to the 2-d benchmarks. When moving from 2-d to 3-d, and increasing the number of metal layers from 2 to 6, via counts approximately triple and account for 50% to 74% of total cost. Given that the resistivity of tungsten (the material of vias) is much higher than that of copper and aluminum, vias are also critical in timing-driven routing. Furthermore, the high variability in via parasitics [27] and the common practice of post-route via doubling to improve yield [17, 18] suggest that via minimization is a key issue in routing at the nanometer scale. In contrast, previous works on negotiated-congestion routing do not consider via minimization because they focus on FPGA routing.

### 5.3 Steiner Trees vs. MSTs

Traditionally net decomposition has been done using Minimal Spanning Tree (MST) algorithms, but fast and extremely accurate Rectilinear Steiner Minimal Tree (RSMT) construction algorithms have become increasingly popular [4, 23, 24]. FGR can use any well-formed net decomposition, so we study how the choice of net decomposition affects FGR's overall results—we compare MST to a combination of FLUTE [6] and FastSteiner [14] that returns the better Steiner tree every time. FGR merges segments of decomposed nets, as described in Section 3.3 and produces non-trivial Steiner trees even when given MST decompositions. The results on the ISPD '07 benchmarks are shown in Table 3. Time taken for decomposition by MSTs or Steiner trees is less than 1 minute on all benchmarks and does not significantly impact runtimes. As expected, routed segment length is smaller when Steiner tree algorithms are used, but using Steiner tree algorithms increases via counts by 1.8% and causes total cost to increase by 0.7%. All evidence we have seen suggests that MST decompositions leave more flexibility than RSMTs, allowing one to avoid some detouring. Prior work has shown that RSMTs for a given set of points can vary widely. Specialized techniques can increase flexibility [3], but FLUTE and FastSteiner do not currently optimize tree flexibility. In addition, Steiner points may inadvertently be placed in congested areas during construction, causing increased congestion and detouring. Congestion-driven Steiner trees could be helpful in this context, but MSTs already provide a good solution and can also be biased to avoid congestion.

### 5.4 Layer Assignment vs. Full 3-d Routing

FGR performs 3-d routing by first flattening the routing instance onto a 2-d grid, routing the new 2-d problem instance, and then converting the 2-d solution into a 3-d solution with layer assignment. FGR is also capable of solving 3-d problems directly by using full 3-d maze routing, and in Table 4 we compare both methods. It is readily apparent that full 3-d routing takes far longer than 2-d routing with layer assignment, most likely because 3-d routing is more complex. On the easiest benchmarks, adaptec4 and newblue2, full 3-d routing takes at least 50% longer, but is able to decrease via counts significantly and in turn improve total cost by 2.0% and 2.1%, respectively. On the other hand, on the benchmarks where FGR with layer assignment cannot find a legal solution within 24 hours, newblue1 and newblue3, full 3-d routing produces solutions with significantly more overflow given the same timeout.

## 6. Conclusions

In this paper we have presented FGR, a high-performance global router for nanometer scale designs. FGR's implementation is very compact—core algorithms and data structures require only 1200 lines of C++ code. FGR outperforms the best results from the ISPD '07 Global Routing Contest, as well as previous literature, in terms of route completion, runtime and total wirelength. In particular, FGR improves upon wirelength produced by BoxRouter and MaizeRouter in March 2007 by 9.9% and 8.4%, respectively. Given that no high-performance routers are open-source today, FGR is likely to boost research in physical design, while also leading to better commercial place-and-route tools [9].

[5] FGR can be stopped much earlier, with only a slight increase in overflows.

| Benchmark | | Layer Assignment | | | | | Full 3-d Routing | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Total ovfl | Segment WL (e5) | Vias (e5) | Total cost | Time (m) | Total ovfl | Segment WL (e5) | Vias (e5) | Total cost | Time (m) |
| adaptec | #1 | 0 | 36.37 | 17.36 | **88.45** | 430 | 1456 | 36.02 | 17.55 | 88.70 | 1453 |
| | #2 | 0 | 33.74 | 18.71 | **89.89** | 64 | 2 | 33.36 | 19.06 | 90.54 | 1444 |
| | #3 | 0 | 97.02 | 34.21 | **199.66** | 243 | 2 | 96.69 | 34.77 | 201.01 | 1487 |
| | #4 | 0 | 91.28 | 30.56 | 182.96 | 55 | 0 | 91.39 | 29.32 | **179.36** | 83 |
| | #5 | 0 | 103.89 | 52.03 | **259.98** | 740 | 5512 | 102.78 | 52.27 | 259.61 | 1462 |
| newblue | #1 | 514 | 24.15 | 23.37 | **94.26** | 1442 | 1012 | 24.21 | 22.33 | 91.19 | 1447 |
| | #2 | 0 | 47.91 | 28.08 | 132.16 | 10 | 0 | 47.93 | 27.15 | **129.40** | 18 |
| | #3 | 39828 | 75.63 | 32.69 | **173.71** | 1501 | 51098 | 75.73 | 29.30 | 163.63 | 1827 |

Table 4. Comparing layer assignment with full 3-d routing on the 3-d instances of the ISPD '07 benchmarks [13]. Total cost of the better of the two solutions (compared first by overflow and then by total cost) for each benchmark are highlighted in bold.

**Challenges for Future Research.** FGR's core algorithms are directly relevant to detail routing of ASICs and FPGAs, while its constraint-driven nature makes it amenable to the handling of complex design rules. To this end, a key challenge for future research is to develop a prototype of a detail routing tool based on negotiated-congestion routing. Such a prototype would be particularly useful to explore design rules and models expected at future technology nodes.

Another key challenge is to integrate accurate congestion modeling provided by FGR into global and detail placement. This could be used to mitigate congestion early and to provide accurate information about length of individual wires, which is particularly important in timing-driven placement.

**Benchmarking Challenges.** Much empirical progress in place-and-route research has been driven by carefully designed benchmark suites. Therefore, ensuring the high quality of public suites is of major importance. To this end, a comparison of FGR results on the ISPD '98 and ISPD '07 benchmarks suggests that the ISPD '98 benchmarks have become outdated. They do not allow for via minimization since they have but one layer of metal, and their sizes are at least an order of magnitude smaller than common industrial designs. In terms of difficulty, 7 of the 10 ISPD '98 benchmarks can be routed with only minor detouring (no more than 1% above Steiner-tree lengths produced by FLUTE [6]) which is in contrast to the 3-d variants of the ISPD '07 benchmarks where FGR produces solutions with at least 2% detouring in all cases and up to 7% for more difficult benchmarks.

The sheer size of the ISPD '07 benchmarks can complicate some empirical studies, however, an easy solution is to partition them and use fractions of their layouts to quickly evaluate changes in routing algorithms. In addition, it has been shown that one can control the difficulty of routing instances through whitespace distribution during placement. Distributing whitespace uniformly generally improves routability, making routing problems easier [1] while congestion-driven allocation further simplifies routing [26].

Neither the ISPD '98 nor the ISPD '07 benchmarks require significant detouring. It is likely that by adding obstacles or zero-capacity GCells one can make routing more difficult. In particular, multi-pin net decomposition in the presence of obstacles remains a challenging problem, as evidenced by recent papers on obstacle-avoiding Steiner-trees [19, 28].

**Acknowledgments.** This work was partially supported by the National Science Foundation (NSF) and the Horace H. Rackham School of Graduate Studies at the University of Michigan.

## REFERENCES

[1] S. N. Adya, I. L. Markov and P. G. Villarrubia, "On Whitespace and Stability in Physical Synthesis," *Integration: the VLSI Journal* 39(4), pp. 340-362, 2006.

[2] C. Albrecht, "Global Routing by New Approximations for Multicommodity Flow," *IEEE TCAD* 20(5), pp. 622-632, 2001.

[3] E. Bozorgzadeh, R. Kastner and M. Sarrafzadeh, "Creating and Exploiting Flexibility in Rectilinear Steiner Trees," *IEEE TCAD* 22(5), pp. 605-615, 2003.

[4] M. Cho and D. Z. Pan, "BoxRouter: A New Global Router Based on Box Expansion and Progressive ILP," In Proc. *DAC*, pp. 373-378, 2006.

[5] M. Cho, H. Xiang, R. Puri and D. Z. Pan, "Wire Density Driven Global Routing for CMP Variation and Timing," In Proc. *ICCAD*, pp. 487-492, 2006.

[6] C. C. N. Chu and Y.-C. Wong, "Fast and Accurate Rectilinear Steiner Minimal Tree Algorithm for VLSI Design," In Proc. *ISPD*, pp. 28-35, 2005.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001.

[8] W. A. Dees, Jr. and P. G. Karger, "Automated Rip-up and Reroute Techniques," In Proc. *DAC*, pp. 432-439, 1982.

[9] R. Goering, "IC Routing Contest Boosts CAD Research," EE Times, March 22, 2007. `http://www.eetimes.com/showArticle.jhtml?articleID=198500084`

[10] R. Hadsell and P. H. Madden, "Improved Global Routing through Congestion Estimation," In Proc. *DAC*, pp. 28-34, 2003.

[11] J. Hu and S. S. Sapatnekar, "A Survey on Multi-net Global Routing for Integrated Circuits," *Integration, the VLSI Journal* 31(1), pp. 1-49, 2001.

[12] ISPD 1998 Global Routing benchmark suite. `http://www.ece.ucsb.edu/~kastner/labyrinth`

[13] ISPD 2007 Global Routing Contest and benchmark suite. `http://www.sigda.org/ispd2007/rcontest/`

[14] A. B. Kahng, I. I. Mandoiu and A. Zelikovsky, "Highly Scalable Algorithms for Rectilinear and Octilinear Steiner Trees," In Proc. *ASP-DAC*, pp. 827-833, 2003.

[15] R. Kastner, E. Bozorgzadeh and M. Sarrafzadeh, "Pattern Routing: Use and Theory for Increasing Predictability and Avoiding Coupling," *IEEE TCAD* 21(7), pp. 777-790, 2002.

[16] Lagrange multipliers. `http://en.wikipedia.org/wiki/Lagrange_multipliers`

[17] K.-Y. Lee and T.-C. Wang, "Post-routing Redundant Via Insertion for Yield/Reliability Improvement," In Proc. *ASP-DAC*, pp. 303-308, 2006.

[18] C.-W. Lin et. al., "Recent Research and Emerging Challenges in Physical Design for Manufacturability/Reliability," In Proc. *ASP-DAC*, pp. 238-243, 2007.

[19] C.-W. Lin et. al., "Efficient Obstacle-avoiding Rectilinear Steiner Tree Construction," In Proc. *ISPD*, pp. 127-134, 2007.

[20] D. McGrath, "Routing Technology Came from Within Cadence, execs say," EE Times, Sept. 8, 2006. `http://www.eetimes.com/showArticle.jhtml?articleID=192700243`

[21] L. McMurchie and C. Ebeling, "PathFinder: A Negotiation-based Performance-driven Router for FPGAs," In Proc. *FPGA*, 1995.

[22] M. Moffitt, Personal communication, March 2007.

[23] M. Pan and C. Chu, "FastRoute: A Step to Integrate Global Routing into Placement," In Proc. *ICCAD*, pp. 464-471, 2006.

[24] M. Pan and C. Chu, "FastRoute 2.0: A High-quality and Efficient Global Router," In Proc. *ASP-DAC*, pp. 250-255, 2007.

[25] H. Ren, D. Z. Pan and P. G. Villarubia, "True Crosstalk Aware Incremental Placement with Noise Map," In Proc. *ICCAD*, pp. 402-409, 2004.

[26] J. A. Roy and I. L. Markov, "Seeing the Forest and the Trees: Steiner Wirelength Optimization in Placement," *IEEE TCAD* 26(4), pp. 632-644, 2007.

[27] L. K. Scheffer, "Physical CAD Changes to Incorporate Design for Lithography and Manufacturability," In Proc. *ASP-DAC*, pp. 768-773, 2004.

[28] Z. Shen, C. C. N. Chu and Y. M. Li, "Efficient Rectilinear Steiner Tree Construction with Rectilinear Blockages," In Proc. *ICCD*, pp. 38-44, 2005.

[29] K. So, "Solving Hard Instances of FPGA Routing with a Congestion-Optimal Restrained-Norm Path Search Space," In Proc. *ISPD*, pp. 151-158, 2007.

[30] P. V. Srinivas et. al., "System and Method for Estimating Capacitance of Wires Based on Congestion Information," U.S. Patent 6519745, filed May 26, 2000, issued Feb. 11, 2003.

[31] D. C. Wang, "Method for Estimating Routability and Congestion in a Cell Placement for Integrated Circuit Chip," U.S. Patent 5587923, filed Sept. 7, 1994, issued Dec. 24, 1996.

[32] J. Westra, C. Bartels and P. Groeneveld, "Probabilistic Congestion Prediction," In Proc. *ISPD*, pp. 204-209, 2004.

[33] J. Westra and P. Groeneveld, "Is Probabilistic Congestion Estimation Worthwhile?", In Proc. *SLIP*, pp. 99-106, 2005.