Multilevel Approach to Full-Chip Gridless Routing*

Jason Cong, Jie Fang, and Yan Zhang
Computer Science Department, UCLA
Los Angeles, CA 90095
cong, jfang, zhangyan@cs.ucla.edu

Abstract

This paper presents a novel gridless detailed routing approach based on multilevel optimization. The multilevel framework with recursive coarsening and refinement in a "V-shaped" flow allows efficient scaling of our gridless detailed router to very large designs. The downward pass of recursive coarsening builds the representations of routing regions at different levels, while the upward pass of iterative refinement allows a gradual convergence to a globally optimized solution. The use of a multicommodity flow-based routing algorithm for the initial routing at the coarsest level and a modified maze algorithm for the refinement at each level considerably improves the quality of gridless routing results. Compared with the recently published gridless detailed routing algorithm using wire planning [1], our multilevel gridless routing algorithm is 3× to 75× faster. We also compared our multilevel framework with a recently developed three-level routing approach [1] and a traditional hierarchical routing approach. Our multilevel algorithm generates better detailed routing results with higher completion rates. To our knowledge, this is the first time that multilevel optimization has been applied to IC routing.

1 Introduction

IC routing is an old and well studied topic. A traditional routing system is composed of two stages: global routing and detailed routing. In global routing, the routing region is partitioned into tiles or channels. A rough route for each net is determined among these tiles to minimize the overall congestion. Next, detailed routing is performed for each tile, where the exact implementation of each net is determined. Variable widths and variable spacings may be used for delay and noise minimization(e.g. see [2]), which implies that the gridless detailed router should not restrict the wires to a predefined uniform grid.

Global routing partitions the entire routing region into tiles and tries to find a tile-to-tile path for each net to guide the detailed routing. Many global routing algorithms use a flat approach by trying to find paths for *all* the nets on the finest tiles. Early routing algorithms use the maze searching algorithm [3, 4] or the line-probe algorithm [5, 6] in a net-by-net approach to find the paths for all nets. Multicommodity flow based algorithms have also been proposed, where the routing problem is modeled as a multi-terminal, multicommodity flow problem. Carden, et al., [7] proposed a flow-based global routing algorithm that has a theoretical bound on the optimal solutions. Albrecht [8] presented a recent work that uses a new approximation algorithm to speed up the flow computation. Another recent work uses a combination of maze searching and iterative deletion, as proposed by Cong and Madden [9], for a performance-driven global routing algorithm. However, all flat approaches have a scaling problem when it comes to large designs. Hierarchical approaches were proposed to handle global routing problems as the problem size increases. Heisterman and Lengauer [10] proposed a hierarchical-based integer programming for global routing. Wang and Kuh [11] use a hierarchical $(\alpha, \beta)^*$ algorithm for the MCM global routing.

The traditional two-level routing approaches, however, have two limitations in current and future VLSI designs. First, future designs may integrate over several hundreds of millions of transistors in a single chip. Traditional two-level design flow may not scale well to handle problems of such size. For example, a $2.5 \times 2.5 cm^2$ chip using a 0.07μm processing technology, as predicted by NTRS'97 [12], may have over 360,000 horizontal and vertical routing tracks at the full chip level. That will translate to about 600×600 routing tiles if we balance the grid size between the global routing and detailed routing stages. This presents a significant challenge to the efficiency of both stages. Even with the three-level routing system [1], which we proposed for deep sub-micron VLSI routing, the routing region has to be partitioned into over 100×100 tiles on both the top-level global routing and intermediate-level wire planning stage (assuming the final tile for detailed routing is about 30×30 tracks for the efficiency of a gridless router). The problem size at each level is still very large. Therefore, as the designs grow, more levels of routing are needed for larger designs. Rather than a predetermined, manual partition of levels which may have discontinuity between levels, an automated flow is needed to enable seamless transitions between the levels.

Moreover, the delay and noise due to the global interconnects should be carefully considered during routing [13]. A timing- and crosstalk-driven detailed router [14] may be too restricted to local optimal solutions due to the lack of a global picture of the performance issues. Several optimization al-

*This research is partially supported by the MARCO/DARPA Gigascale Silicon Research Center (GSRC) and the National Science Foundation under the award CCR-0096383.

gorithms are proposed at the global routing level, such as a performance-driven global router using high-performance routing topologies and optimal wire sizing [9], and a noise-constrained wire spacing and track assignment algorithm for global routing refinement [15, 16]. Although each algorithm tries to optimize one or two performance aspects on its own, these optimization algorithms are implemented as separated modules and interleaving them in the current routing flow is difficult. A simple integration of putting them one after another may not get the best overall performance on the final layouts.

In this paper, we propose a novel multilevel routing framework for the gridless routing problem of large VLSI designs. The multilevel methods was originally used as a means of accelerating numerical algorithms for partial differential equations (e.g. [17, 18]). In this past decade, it has been also applied to other areas, such as image processing, combinatorial optimization, control theory, statistical mechanics, quantum electrodynamics, and linear algebra. Multilevel techniques for VLSI physical designs have shown promising results recently. Good progress has been made in multilevel circuit partitioning and placement. The multilevel partitioning algorithm *hMETIS* [19] produces the best cut size minimization in circuit partitioning. The multilevel performance-driven partitioning algorithm *HPM* [20] produces the best balance of delay and cut size minimization results for circuit partitioning. The multilevel placement algorithm *mPL* [21] achieves comparable circuit placement quality with the well known *GORDIAN* package [22] with over 10$\times$ speed-up on designs with over 200K movable objects. These successes led us to investigate the application of the multilevel scheme to handling large VLSI routing problems.

Our multilevel framework features an iterative coarsening algorithm and an iterative refinement algorithm in a "V-shaped" flow, which is typical for multilevel optimization (see Figure 1). On the downward pass, the design is recursively coarsened, and an estimation of routing resources is calculated at each level. At the coarsest level, a multicommodity flow algorithm is used to generate an initial routing result. On the upward pass, a modified maze searching algorithm is carried out iteratively to refine the results from level to level. The final results of the multilevel planning algorithm are tile-to-tile paths for all the nets. These paths are used to guide a gridless detailed routing algorithm to find the exact connection for each net. The hierarchical nature of the multilevel scheme makes it very scalable to large designs. Moreover, the iterative refinement process provides a framework for seamless integration of different algorithms on different levels. Our experimental results show that the multilevel flow improves routability over the traditional two-step approach of global routing followed by detailed routing and the hierarchical approaches.

Section 2 provides an overview of our multilevel routing framework. Section 3 explains our tile partitioning and resource estimation algorithm. A coarsening process generates the level representations from the finest level to the coarsest level. A multicommodity flow algorithm is used to compute the initial routing solution at the coarsest level in Section 4. A modified maze searching algorithm is proposed in Section 5 for un-coarsening and refinement in the multilevel routing flow. Finally, the effectiveness of our algorithm is validated with experimental results in Section 6. The paper concludes with a discussion of several possible extensions of the proposed multilevel framework for VLSI routing in Section 7.

2 Overview

Figure 1 illustrates our multilevel framework for VLSI routing. The routing area is partitioned into routing tiles. Our algorithm goes through multilevel planning to find a tile-to-tile path for each net among these tiles. In contrast, most traditional global routing algorithms [7, 8, 9] try to directly find routing solutions on the finest tiles. For large designs, the number of tiles may be too great for these algorithms to handle. Our multilevel approach first accurately estimates the routing resource using a line-sweeping algorithm on the finest tile level. A recursive coarsening process is then employed to merge the tiles and build coarser and coarser level representations (Section 3). At each coarsening stage, the resource of each tile is estimated from the previous finer level tiles forming the current tile. This coarsening process is known in multilevel literature as the "downward pass." Once the coarsening process has reduced the number of tiles to below a certain threshold, the initial routing is computed using a multicommodity flow based algorithm (Section 4). The initial routing result is refined in the reverse direction of coarsening, known as the "upward pass," by a modified maze searching algorithm (Section 5). When the final tile-to-tile paths are found at the finest level of tiles for all the nets, a gridless detailed routing algorithm [23] is applied to find the exact path for each net.

Compared to the flat approaches, the multilevel algorithm is much more scalable to large designs. Traditionally, hierarchical approaches [10, 11] are used to overcome the scaling problem on large designs. These methods also build multilevel hierarchical representations of the routing region, however, from the coarsest level to the finest level. The key problem with the hierarchical approaches is the lack of detailed routing information available to make routing decisions at the coarse level, while these coarse-level decisions *constrain* the fine-level solutions. Thus, if an unwise decision is made at any level, it is almost impossible or very costly to revise it at a finer level in hierarchical approaches, as illustrated in Figure 2. The "V-shaped" flow of a multilevel approach is more flexible than a hierarchical approach. The uncoarsening pass allows the fine level router to *refine* the coarse level result. The coarse level solution only provides a *guide* (as opposed to a *constraint* in the case of hierarchical routing) to fine level path searching. That is the fine level path searching algorithm has the flexibility to deviate from the coarse level path when more detailed information about local resource and congestion is considered. This feature makes the multilevel method converge to better solutions with higher efficiency.

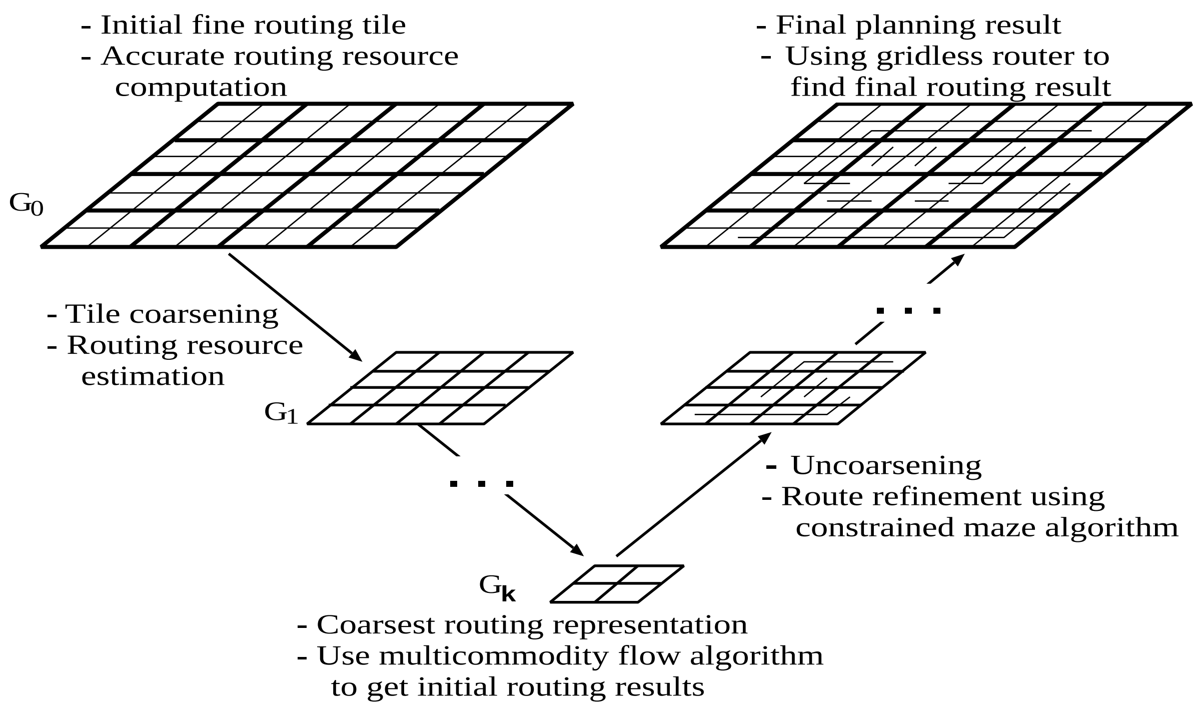
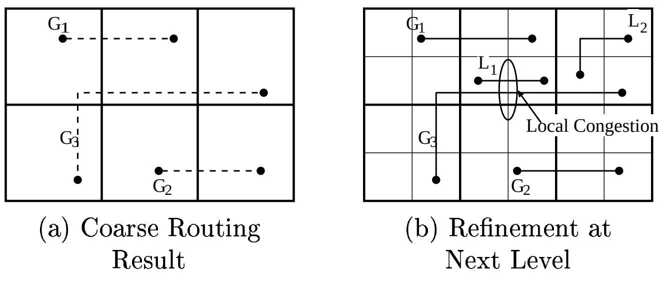

- Initial fine routing tile
- Accurate routing resource computation
G0
- Tile coarsening
- Routing resource estimation
G1
Gk
- Coarsest routing representation
- Use multicommodity flow algorithm to get initial routing results
- Uncoarsening
- Route refinement using constrained maze algorithm
- Final planning result
- Using gridless router to find final routing result

Figure 1: Multilevel Routing Flow



(a) Coarse Routing Result

(b) Refinement at Next Level

Figure 2: Limitation of Hierarchical Approaches. In a hierarchical approach, coarse-level decisions are based on limited information, as shown in (a). However, fine-level refinement lacks the flexibility to change the coarse-level results. Thus, we have a local congestion after the refinement, as shown in (b).

3 Routing Resource Estimation and Generation of Multilevel Hierarchy

The first step in our multilevel flow is to build up the multiple levels of routing region representations along the "downward pass." The routing region is first partitioned into an array of fine tiles, each with the same height and width. We denote this level as level 0. We then build a three-dimensional routing graph, denoted G_0, such that each node in G_0 represents a tile in some routing layer in level 0. Every two nodes in G_0 that represent a pair of neighboring tiles are connected with an edge. The edge capacity represents the routing resources at the common boundary of the two tiles. The ultimate objective of the multilevel routing algorithm is to find a tile-to-tile path for each net in G_0 to be used to guide our gridless detailed router for searching a connection for each net.

The multilevel algorithm first accurately estimates the routing resources at the finest level, then recursively coarsens the representations on different levels in a downward pass. The resources at the finest level is estimated as follows. Due to the gridless nature of our routing problem, we use actual dimensions of the obstacles to compute the routing resources. For each tile boundary, we use a line-sweeping algorithm, similar to the estimation algorithm used in [1], to compute its capacity C. The sweeping algorithm cuts the routing region into horizontal (or vertical) empty rectangles called *slices*. Let W_i and D_i be the width and depth of each slice S_i, as shown in Figure 3. Suppose that the tile depth is D. Then, the boundary capacity can be computed as a *weighted* sum of widths of empty slices along the tile boundary computed by the following formula:

$$C = \sum_i W_i \times D_i / D \qquad (1)$$

The inter-layer edge capacity of the routing graph G_i represents the available resources for vias, and is computed as the sum of the areas of all empty slices intersections between the two tiles connected by the edge.

Given the accurate routing capacity estimation at the finest level and the grid graph G_0 that stores such information, the coarsening process for our multilevel routing is actually quite straightforward. At a coarser level (level i+1), the tiles are built from the finer level tiles (level i) by merging neighboring tiles. The coarse level graph, G_{i+1}, which represents the routing resources available at the coarse tiles, can also be de-

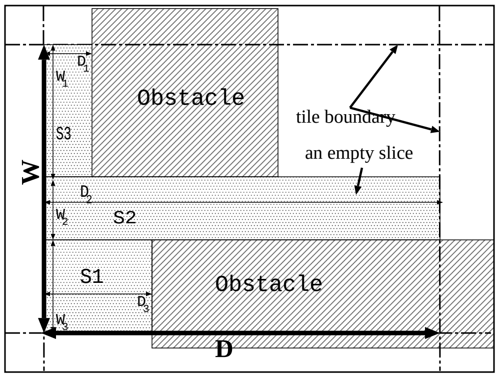


Figure 3: Resource Estimation in Initial Tiles

rived from the fine level graph G_i directly. The capacity of an edge (u_{i+1}, v_{i+1}) on G_{i+1} is the sum the capacities of the edges in G_i that connect the tiles merged into u_{i+1} and the tiles merged into v_{i+1}. We iteratively coarsen the tiles and the routing graphs until the size of the graph falls below a predetermined threshold.

4 Initial Routing Using Multicommodity Flow Algorithm

After the routing tiles are coarsened to a certain level, the coarsening process stops. A set of tile-to-tile paths are computed for the nets crossing the coarsest tile boundaries. This process is called the *initial* routing, which is quite important to the final result of multilevel routing. First, long interconnects that span multiple tiles are among the nets in initial routing. Normally, these long interconnects are timing-critical and may also suffer noise problems due to coupling along their paths. Initial routing algorithms should be capable of handling these performance issues. Second, the initial routing result will be carried all the way down to the finest tiles through the refinement process in the multilevel scheme. Although a multilevel framework allows finer level designs to change coarser level solutions, a bad initial routing solution will slow down the refinement process and may even degrade the final solution.

In our approach, we use a multicommodity flow based algorithm to compute the initial routing solution at the coarsest level. Several existing routing algorithms use the flow-based algorithm (e.g., [7, 8]). We chose the flow-based algorithm rather than a net-by-net approach using a maze-searching algorithm or an iterative-deletion algorithm for several reasons. First, the flow algorithm is fast enough for a relatively big grid size. Although in theory we can continue coarsening the tiles so that the number of tiles is very small, the coarsest level should have reasonable granularity so that the initial routing results can influence the later refinement process. Second, the flow algorithm considers all the nets at the same time. This removes part of the net ordering problem in the net-by-net approaches. A globally good solution for *all* nets is especially important at the coarsest level because its solution will be carried to influence all finer-level solutions through the refinement process. Last, a flow algorithm can be integrated with other optimization algorithms to consider special requirements of certain critical nets. For example, we can compute high-performance tree structures using an A-Tree algorithm [24] to compute minimum Steiner trees [25] or a RMP algorithm [26] to compute tree structures with buffers. These tree structures can potentially be used as initial solutions for the flow algorithm.

The objective of the initial routing is to minimize the congestion on the routing graph G_0, which represents the coarsest tiles. Our current implementation considers only two-pin nets. Multi-terminal nets are first partitioned by a heuristic algorithm. We first compute a set of candidate paths for each net on the coarsest tile graph G_k. For a given net i, let $\mathbf{P}_i=\{P_{i,1}, \ldots, P_{i,l_i}\}$ be the set of possible paths. Our current implementation does not consider delay minimization, and focuses mainly on routability and wirelength optimization. Therefore, we use only the shortest paths as candidates for each net. Assume the capacity of each edge on the routing graph is $c(e)$, and $w_{i,e}$ is the cost for net i to go through edge e. Let $x_{i,j}$ be an integer variable with possible values 1 or 0 indicating if path $P_{i,j}$ is chosen or not ($1 \leq j \leq l_i$). Then, the initial routing problem can be formulated as as a mixed integer linear programming problem as follows:

$$\begin{array}{lcccl} & & min\ \lambda & & \\ subject\ to & & & & \\ \sum_{e\,\in\,P_{i,j}} w_{i,e}x_{i,j} & \leq & \lambda\ c(e) & for & e \in E \\ \sum_{j=1}^{l_i} x_{i,j} & = & 1 & for & i=1,\ldots,n_k \\ x_{i,j} & \in & \{0,1\} & for & i=1,\ldots,n_k; \end{array} \qquad (2)$$

where n_k is the number of nets to be routed at level k. Normally, this mixed integer programming problem is relaxed to a linear programming problem by replacing the last constraint in Equation 2 with:

$$x_{i,j} \ \geq\ 0 \quad for \quad \begin{array}{l} i=1,\ldots,n_k; \\ j=1,\ldots,n_k; \end{array} \qquad (3)$$

After relaxation, a maximum flow approximation algorithm can be used to compute the fraction value of $x_{i,j}$ for each net in the above linear programming formulation. Traditionally, the maximum flow approximation algorithm picks a path and routes a unit flow along the path. Then it multiplies the length of every edge on this path by $1+\epsilon$ with a fixed ϵ. Our implementation of fraction computation is faster because it uses the approximation method proposed in [8]. In this algorithm, a maximum s-t flow is computed using a faster and more straightforward method: after picking a path, instead of routing the path with a unit flow, we increase the flow along the path as much as possible to saturate the minimum capacity edge along the path. Garg and Konemann [27] proved the optimality of this method and gave a detailed explanation of its application to multicommodity flow computation.

After the fractional result for each path are computed, we need to map the fractional results to integer results. Our algorithm calls a randomized rounding algorithm to convert the fractional values into 0 or 1 values for candidate paths of each net so that one path is chosen for each net. The randomized rounding approach for global routing was first used in [28] and an error bound was estimated. Using this simple heuristic, we can quickly get an integral solution. This method does not guarantee that there is no overflow at the tile boundaries. In general, overflow can be corrected by ripup and re-route. Our implementation, however, does not use ripup and re-route at this level, because congestion is not a significant problem, and we rely on subsequent refinement steps to remove the possible overflow.

Our experimental results show that the flow-based initial routing algorithm is more efficient than a maze routing approach, in terms of both the final completion rate and the runtime. The results are shown in Table 2 in Section 6.

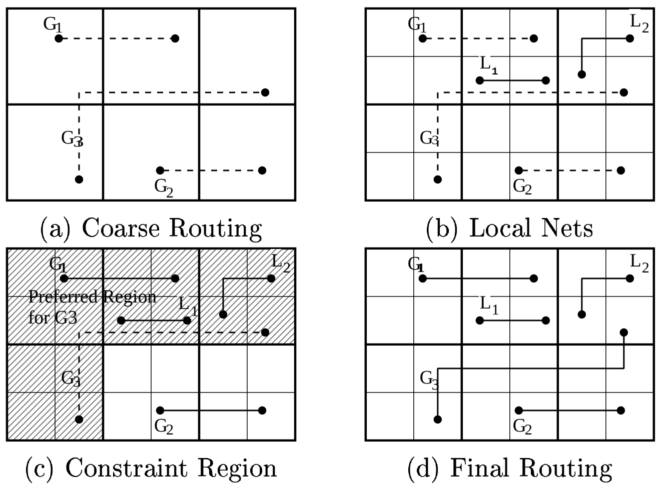
5 Incremental Refinement Algorithm for Multilevel Framework

One major difference between the hierarchical routing and multilevel routing approaches is that a multilevel framework allows the finer level to *change* coarser-level routing solutions. In the upward pass of the multilevel framework, paths computed by the initial flow-based algorithm are refined from level to level until the finest tiles are finally reached.

There are two types of nets during race step of the refinement. One type is "new" nets that just appear at the current level, as shown by solid lines in Figure 4 (b). These nets are relatively short and do not cross coarser tile boundaries, thus, they are not modeled at coarser levels. We call them *local nets*. New paths need to be created for these nets at the current level. Another set of nets are those carried over from the previous coarser-level routing. We need to refine the tile-to-tile paths for these nets at the current level.

Finding paths for the local nets is relatively easy as they are short and each net crosses at most two tiles. Thus, during each refinement stage, we determine paths for these nets prior to coarser level path refinements. Furthermore, routing these nets before any refinement gives a more accurate estimation of local resources.

The major part of the refinement work comes from refining those coarser-level nets. In general, the amount of work needed for refinement depends on the quality of the coarse level solution. In one extreme, the choices of the paths at the coarse level are also optimal at the current level. We only need to refine the paths within the regions defined by the paths in coarse tiles. In this case, the multilevel algorithm is the same as a hierarchical algorithm. On another extreme, when the coarser solution is totally useless or misleading, the best we can do at the current level is to discard the coarser level solution and search for a new path for each coarse level net among all finer tiles at this level. In this case, we end up doing full planning at the current level. We believe that the reality lies between these two extreme cases. A good coarse level routing solution provides some good hints as to where the the best solution might be. However, if we restrict our



(a) Coarse Routing (b) Local Nets (c) Constraint Region (d) Final Routing

Figure 4: Constrained Maze Refinement. Previous routing at coarse grid is shown in (a). Local nets at current level are routed first. Preferred regions are defined by path at coarse tile, as shown in (c). However, not restricted to higher-level results, the modified maze algorithm can change the upper-level tile constraints by seeing the local congestion and arrive at better solutions (d).

search space for the finer tile path to be totally within coarse level tiles planned in the previous level, as in a hierarchical approach, we will lose the flexibility to correct the mistakes made by coarser levels.

In order to keep the coarser-level guide while still maintaining the flexibility to search for all finer tiles, we have implemented a net-by-net refinement algorithm using a modified maze-searching algorithm. We use the path on coarser tiles as a guide to search for a path at the current level. A *preferred region* is defined as the set of tiles that the coarse level path goes through. Weights and penalties associated with each routing graph edge are computed based on the capacities and usage by routed nets. Additional penalties are assigned to graph edges linking to and going between the graph nodes corresponding to tiles that are not located within the preferred region, as shown in Figure 4 (c). Dijkstra's shortest path algorithm [29] is used to find a weighted shortest path for each net, considering wire length, congestion, and the coarser-level planning results. In general, Dijkstra's algorithm may be slow in searching for a path in a large graph. However, by putting penalties to non-preferred regions, we can guide the path to search within the preferred regions first. The penalty is chosen so that it does not prevent the router from finding a better solution that does not fall into the preferred region. Figure 4 (d) shows an example where, with more accurate finer-level tile information and local nets information, the modified maze routing algorithm finds a path for net G_3 that is not totally within its preferred region.

With the upward pass refinement process, we can get a globally optimized solution for each net on the finest tiles. Finally, we use the gridless detailed routing engine presented

Table 1: Examples Used for Multilevel Routing

Ex.	Size (μm)	Layer	#Nets	#Cells	#Pins
Mcc1	39000 ×45000	4	1694	MCM	3101
Mcc2	152400 ×152400	4	7541	MCM	25024
Raytheon	2760 ×1560	4	430	MCM	950
Struct	4903 ×4904	3	3551	n/a	5471
Primary1	7552 ×4988	3	2037	n/a	2941
Primary2	10438 ×6468	3	8197	n/a	11226
S5378	4330 ×2370	3	3124	1659	4734
S9234	4020 ×2230	3	2774	1450	4185
S13207	6590 ×3640	3	6995	3719	10562
S15850	7040 ×3880	3	8321	4395	12566
S38417	11430 ×6180	3	21035	11281	32210
S38584	12940 ×6710	3	28177	14716	42589

Table 2: Comparison of Flow-Based with Maze-Based Initial Routing Method

Ex.	Maze-Based Initial Routing			Flow-Based Initial Routing	
	Run Time(s)	# Rtd. Nets	Cmp. Rates	Run Time(s)	Cmp. Rates
Mcc1	514.8	1672	98.7%	436.7	99.4%
Mcc2	8116.1	7463	99.0%	7644.8	99.1%
Struct	274.6	3551	100%	316.8	100%
Prim1	335.6	2037	100%	350.2	100%
Prim2	2546.1	8189	99.9%	2488.4	100%
S5378	66.4	2940	94.1%	54.0	94.8%
S9234	65.1	2536	91.4%	41.0	92.3%
S13207	290.3	6501	92.9%	188.8	94.0%
S15850	691.4	7887	94.8%	403.4	94.5%
S38417	921.7	19411	92.3%	733.6	93.3%
S38584	2237.4	26271	93.2%	1721.6	94.1%
avg.			96.0%		96.5 %

in [23] to find the final connection for each net under the guide of the tile-to-tile paths found by our multilevel routing algorithm.

6 Experimental Results

We have implemented our multilevel routing system including recursive coarsening, a multicommodity flow-based initial routing algorithm, a modified maze-routing algorithm for incremental refinement and a gridless detailed router. The multilevel routing results are finalized using the efficient multilayer gridless routing engine presented in [23]. One major enhancement over the method in [23] is that instead of building up a connection graph for the whole routing area (the planned path corridor), we store a portion of the routing graph grids at each tile, and combine them for each net on the fly. In this way, we significantly reduce the size of the connection graph (i.e., the non-uniform grid defined in [23], and thus the total runtime.

We have tested our multilevel routing scheme on a wide range of benchmarks, including MCM examples and several standard cell examples (Table 1). Mcc1, Mcc2 and Raytheon are MCM examples, where Mcc1 has 6 modules, Mcc2 has 56 modules, and Raytheon has 133 modules (a mixed signal MCM implementation of a high-speed modem). Our experimental results are collected on a Sun Ultra-5 440Mhz with 384MB of memory. Notice that in the tables, "Ex." stands for examples, "Rtd. Nets" for routed nets, "Cmp. Rates" for completion rates, and "avg." for average.

First, we evaluated the impact of using the flow-based initial routing algorithm. Table 2 compares the multilevel routing scheme using a flow-based initial routing algorithm with the one using maze-based initial routing algorithm. It shows that using the flow-based initial routing method can achieve better completion rate as well as less runtime. Note that the router will try to search a larger space for solution of a failed net, which will increase the runtime, so improving the completion rate often speeds up the algorithm.

We then compared our multilevel routing scheme with a simple net-by-net gridless detailed router [23] and the recently presented gridless detailed routing algorithm with wire planning [1]. Table 3 compares the runtimes and completion rates for these three approaches. The experimental results show that the multilevel scheme results in a 3× to 75× speedup in runtime while generating routing results with higher completion rates. Note that this experiment only included the smallest examples from Table 1, since neither the simple net-by-net approach nor the wire planning approach can scale to handle larger examples. Their runtimes are very long, and the memory consumption is high for large designs.

Furthermore, we compared our multilevel routing algorithm with the three-level routing flow recently presented at ISPD'2000 [1]. The three-level flow features a performance-driven global router [9], a noise-constrained wire spacing and track assignment algorithm [16], and finally a gridless detailed routing algorithm with wire planning [1]. In this experiment, the global router partitions each example into 16 × 16 routing tiles. Nets crossing the tile boundaries are partitioned into subnets within each tile. After the pin assignment, the gridless detailed routing algorithm routes each tile one by one. In Table 4, we report the total runtimes as well as the time spent on each of these three steps. We use a separate program to merge the subnets in different tiles back to the original nets. We count both the lower bound and the upper bound of the completion rate. When calculating the lower bound, one subnet failure will lead to the failure of all two-pin nets in that global net; and when calculating the upper bound, one subnet failure will lead to the failure of only one two-pin net in that

Table 5: Experiment Results of Hierarchical with Ripup and Replan and Multilevel Routing

Ex.	Hierarchical Routing with Ripup and Replan			Multilevel Routing	
	Run Time(s)	# Rtd. Nets	Cmp. Rates	Run Time(s)	Cmp. Rates
Mcc1	947.9	1600	94.5%	436.7	99.4%
Mcc2	10101.4	7161	95.6%	7644.8	99.1%
Struct	324.5	3551	100%	316.8	100%
Prim1	353.0	2037	100%	350.2	100%
Prim2	2423.8	8194	100%	2488.4	100%
S5378	57.9	2964	94.9%	54.0	94.8%
S9234	40.7	2564	92.4%	41.0	92.3%
S13207	161.9	6540	93.5%	188.8	94.0%
S15850	426.1	7874	94.6%	403.4	94.5%
S38417	754.6	19596	93.2%	733.6	93.3%
S38584	1720	26461	93.9%	1721.6	94.1%
avg.			95.7%		96.5%

global net. Our results show that multilevel routing achieves higher completion rates using shorter runtimes (especially for large designs).

Another approach to handling large designs is to use a hierarchical routing flow followed by a ripup and replan. As discussed in Section 2, the difference between a multilevel approach and a hierarchical approach is that coarse level results in the hierarchical approach constrain the fine level results. We modified our multilevel flow and made it a hierarchical approach with ripup and replan. Table 5 compares the routing results of such a hierarchical approach with those of the multilevel approach. Although the hierarchical approach gains a little bit in runtime in some cases, by constraining the search spacing during the un-coarsening process, it loses to our multilevel routing in terms of completion rate. This trend is especially true in designs with many global nets, such as Mcc1 and Mcc2, which means that the multilevel planning method can generate planning results with better quality.

7 Conclusion and Future Work

We present a novel routing system using a multilevel method. It has several advantages. First, it scales well on larger designs. The partition of routing regions provides a natural hierarchy for the routing levels. The downward pass of recursive coarsening builds the representations of routing regions at different levels. When the designs become larger, additional levels can be added in the multilevel framework, while the overall routing flow is preserved. Second, this multilevel method provides a good framework for integrating different routing algorithms and allows different algorithms to be used on different levels. In our approach, we used a flow-based algorithm to compute the initial routing results and a modified maze-searching algorithm to iteratively refine the results. The multilevel framework also enables a seamless interaction of the routing results at different levels. Coarse level results only influence rather than restrict fine level path searchings. This is especially important because the iterative refinements with more and more detailed information are critical for achieving both the performance and quality of the final routing results. In our experimental results, we showed that our multilevel framework gives a $3\times$ to $75\times$ speedup over a gridless detailed routing [1], allowing us to scale to large VLSI routing examples. Compared to a classical hierarchical routing flow and a recent three-level routing flow consisting of a global router, a pin assignment algorithm and a detailed router, our multilevel router provides better routability results. This is due to its ability of continuous optimization across multiple levels, as well as the ability of integrating different algorithms in the flow.

There are still several challenges in the proposed multilevel routing framework. First, we need to understand how to integrate different performance optimization algorithms in the framework, allowing them to generate a performance-driven routing flow. Still unknown to us is at which level to insert individual optimization algorithms and how those algorithms will interact with each other throughout the framework. Further research is also needed to integrate multilevel routing with multilevel partitioning [19, 20] and multilevel placement [21].

References

[1] J. Cong, J. Fang, and K. Khoo, "DUNE: A multi-layer gridless routing system with wire planning," in *Proc. International Symposium on Physical Design*, pp. 12–18, Apr. 2000.

[2] J. Cong, L. He, C.-K. Koh, and P. Madden, "Performance optimization of VLSI interconnect layout," *Intergration, the VLSI Journal*, vol. 21, no. 1-2, pp. 1–94, 1996.

[3] S. Akers, "A modification of Lee's path connection algorithm," *IEEE Trans. on Computers*, vol. EC-16, pp. 97–98, Feb. 1967.

[4] J. Soukup, "Fast maze router," in *Proc. 15th Design Automation Conference*, pp. 100–102, 1978.

[5] K. Mikami and K. Tabuchi, "A computer program for optimal routing of printed ciurcuit connectors," *IFIPS Proc*, vol. H-47, pp. 1475–1478, 1968.

[6] D. Hightower, "A solution to line routing problems on the continuous plane," in *Proc. IEEE 6th Design Automation Workshop*, pp. 1–24, 1969.

[7] R. Carden, J. Li, and C.-K. Cheng, "A global router with a theoretical bound on the optimal solution," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 208–216, Feb. 1996.

[8] C. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow," in *Proc. International Symposium on Physical Design*, pp. 19–25, Mar. 2000.

[9] J. Cong and P. Madden, "Performance driven multi-layer general area routing for PCB/MCM designs," in *Proc. 35th Design Automation Conference*, pp. 356–361, Jun. 1998.

[10] J. Heisterman and T. Lengauer, "The efficient solution of integer programs for hierarchical global routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, pp. 748–753, Jun. 1991.

[11] D. Wang and E. Kuh, "A new timing-driven multilayer mcm/ic routing algorithm," in *Proc. IEEE Multi-Chip Module Conference*, pp. 89–94, Feb. 1997.

[12] Semiconductor Industry Association, *National Technology Roadmap for Semiconductors*, 1997.

Table 3: Comparison of Multilevel Routing with Net-by-Net and Wire Planning Detailed Routing

Examples	Net-by-Net DR			WirePlan+DR			Multilevel Routing		
	Run Time(s)	# Rtd. Nets	Cmp. Rate	Run Time(s)	# Rtd. Nets	Cmp. Rate	Run Time(s)	# Rtd. Nets	Cmp. Rate
Mcc1	14216.4	1489	87.9%	5441.2	1540	90.9%	436.7	1683	99.4%
Raytheon	451.1	409	95.1%	346.2	412	95.8%	94.4	417	97.0%
S5378	1153.9	2880	92.2%	4068.0	2855	91.4%	54.0	2963	94.8%
S9234	706.7	2526	91.1%	2644.4	2554	92.1%	41.0	2561	92.3%

Table 4: Comparison of Three-Level Routing with Multilevel Routing

Ex.	Three-Level Routing						Multilevel Routing		
	Run Time (s)				# Rtd.	Cmp.	Run Time	# Rtd.	Cmp.
	G.R.	P.A.	D.R.	Total	Nets	Rates	(s)	Nets	Rates
Mcc1	639.0	25.0	269.2	933.2	978/1499	57.7%/88%	436.7	1683	99.4%
Mcc2	2947.0	166.0	9220.6	12333.6	5233/5451	69.4%/72.3%	7644.8	7474	99.1%
Struct	267.0	6.0	133.2	406.2	3497/3530	98.5%/99.4%	316.8	3551	100%
Prim1	101.0	3.0	135.1	239.1	1983/2018	97.3%/99.0%	350.2	2037	100%
Prim2	480.0	11.0	826.0	1311	7555/8109	92.2%/98.9%	2488.4	8196	100%
S5378	304.0	5.0	121.2	430.2	1581/2607	50.6%/83.4%	54.0	2963	94.8%
S9234	277.0	3.0	75.2	355.2	1686/2467	60.8%/88.9%	41.0	2561	92.3%
S13207	658.0	11.0	430.5	1099.5	3951/6118	56.5%/87.5%	188.8	6574	94.0%
S15850	791.0	16.0	662.1	1469.1	4902/7343	58.9%/88.2%	403.4	7863	94.5%
S38417	1403.0	29.0	2181.9	3560.9	13510/19090	64.2%/90.8%	733.6	19636	93.3%
S38584	3161.0	48.0	3877.5	7086.5	18539/25642	65.8%/91.0%	1721.6	26504	94.1%
avg.						87.8%/89.8%			96.5%

[13] J. Cong and D.-Z. Pan, "Interconnect estimation and planning for deep submicron designs," in *Proc. 36th Design Automation Conference*, pp. 507–510, Jun. 1999.

[14] H.-P. Tseng, L. Scheffer, and C. Sechen, "Timing and crosstalk driven area routing," in *Proc. 35th Design Automation Conference*, pp. 378–381, Jun. 1998.

[15] C. Chang and J. Cong, "Cross talk noise control in gridless general-area routing," in *Proc. ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU)*, pp. 117–122, Mar. 1999.

[16] C. Chang and J. Cong, "Pseudo pin assignment with crosstalk noise control," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, pp. 598–611, Mar. 2001.

[17] A. Brandt, "Multi-level adaptive solution to boundary value problems," *Mathematics of Computation*, vol. 31, no. 138, pp. 333–390, 1977.

[18] W. Briggs, *A Multigrid Tutorial*. SIAM, 1987.

[19] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Applications in VLSI domain," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 7, pp. 69–79, Mar. 1999.

[20] J. Cong, S. Lim, and C. Wu, "Performance driven multilevel and multiway partitioning with retiming," in *Proc. 37th Design Automation Conference*, pp. 274–279, Jun. 2000.

[21] T. Chan, J. Cong, T. Kong, and J. Shinnerl, "Multilevel optimization for large-scale circuit placement," in *Proc. IEEE International Conference on Computer Aided Design*, pp. 171–176, Nov. 2000.

[22] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN : VLSI placement by quadratic programming and slicing optimization," *IEEE Trans. on Computer-Aided Design*, pp. 356–365, Mar. 1991.

[23] J. Cong, J. Fang, and K. Khoo, "An implicit connection graph maze routing algorithm for ECO routing," in *Proc. International Conference on Computer Aided Design*, pp. 163–167, Nov. 1999.

[24] J. Cong, A. Kahng, and K. Leung, "Efficient algorithms for the minimum shortest path Steiner arborescence problem with applications to VLSI physical design," *IEEE Trans. on Computer-Aided Design*, vol. 17, pp. 24–39, Jan. 1999.

[25] M. Borah, R. Owens, and M. Irwin, "An edge-based heuristic for Steiner routing," *IEEE Trans. on Computer-Aided Design*, vol. 13, pp. 1563–1568, Dec. 1994.

[26] J. Cong and X. Yuan, "Routing tree construction under fixed buffer locations," in *Proc. 37th Design Automation Conference*, pp. 379–384, Jun. 2000.

[27] N. Garg and J. Konemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," in *Proc. Annual Symposium on Foundations of Computer Science*, pp. 300–309, Nov. 1998.

[28] R. Karp, F. Leighton, R. Rivest, C. Thompson, U. Vazirani, and V. V. Vazirani, "Global wire routing in two-dimensional arrays," *Algorithmica*, vol. 2, pp. 113–129, 1987.

[29] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.