An Enhanced Multilevel Routing System*

Jason Cong, Min Xie, Yan Zhang
Computer Science Department, UCLA
Los Angeles, CA 90095
cong, xie, zhangyan@cs.ucla.edu

Abstract

In this paper, we present several novel techniques that make the recently published multilevel routing scheme [19] more effective and complete. Our contributions include: (1) resource reservation for local nets during the coarsening process, (2) congestion-driven, graph-based Steiner tree construction during the initial routing and the refinement process and (3) multi-iteration refinement considering the congestion history. The experiments show that each of these techniques helps to improve the completion rate considerately. Compared to [19], the new routing system reduces the number of failed nets by 2× to 18×, with less than 50% increase in runtime in most cases.

1. Introduction

The continuous increase of the problem size of IC routing has become a great challenge to existing routing algorithms. The traditional method for handling the large problem size is to "divide-and-conquer," which breaks the routing problem into two successive steps, global routing and detailed routing.

Global routing partitions the entire routing region into tiles or channels, and tries to find a tile-to-tile path for each net with congestion and performance optimization. There are two kinds of global routing algorithms. Sequential methods route the nets one-by-one in a predetermined order, using either the maze searching algorithm [1][2] or the line-probe algorithm [3][4]. However, the solution quality is often affected by the routing order. Iterative methods try to overcome the net ordering problem by performing multiple iterations. The negotiation-based iterative global routing scheme was proposed in [5], and later on used in FPGA routing [6]. The flow-based iterative methods were also proposed [7][8], where the global routing problem is modeled as a multi-terminal, multi-commodity flow problem and approximate solutions are computed iteratively. A more recent work used a combination of maze searching and iterative deletion for solving the performance-driven global routing algorithm [9].

After global routing is completed, detailed routing is performed within each tile or channel, where the exact geometric layout of all nets is determined. There are two types of detailed routing approaches, grid-based or gridless routing. A gridless detailed router allows arbitrary widths and spacings for different nets, which can help to optimize the circuit performance and to reduce noise [10][11]. However, the design size that a gridless router can handle is usually limited, due to the high complexity of the routing problem.

Since most global routing algorithms run directly on a 2-D or 2.5-D array of routing tiles, such flat approaches may not scale well to large designs. In [12], a 3-level routing scheme with an additional wire-planning phase between the performance-driven global routing and the detailed routing was proposed. The additional planning phase improved both the completion rate and the runtime.

Hierarchical approaches with multiple levels of hierarchy have also been used to handle large routing problems. The first hierarchical method [13] was proposed for channel routing by Burstein. Heisterman and Lengauer proposed a hierarchical integer programming-based algorithm for global routing [14]. Wang and Kuh proposed a hierarchical $(\alpha, \beta)^*$ algorithm [15] for the MCM global routing. The problems with the hierarchical approaches are: (1) the higher level solutions will constrain the lower level solutions, and (2) the lack of detailed information at the higher levels makes it difficult to make good/well-informed decisions at the higher levels. Therefore, when an unwise decision is made at some point, it would be very costly (through rip up and reroute) to revise it later at a finer level.

The first multilevel routing framework was proposed in [19], inspired by the recent successes of the multilevel technique in VLSI physical designs, including multilevel circuit partitioning [16][17] and multilevel placement [18]. The experimental results showed that this multilevel router, named MRS in this paper, is 1.5× to 15× faster than the 3-level routing system [12] with a 6.7% average completion rate improvement.

In this paper, we present an enhanced multilevel router, named MARS (*Multilevel Advanced Routing System*), which incorporates several new techniques to improve the quality of the multilevel routing algorithm in [19], including resource reservation, a graph-based Steiner tree heuristic and a history-based multi-iteration scheme.

The rest of the paper is organized as follows. In Section 2, we review the MRS router in [19] and discuss its

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

0-7803-7607-2/02/$17.00 ©2002 IEEE

limitations. In Section 3, we describe the MARS router with new techniques used in different multilevel planning phases. Section 4 presents comprehensive experimental evaluations of the individual algorithms as well as the overall performance of the MARS router. In Section 5, we conclude our paper with some discussion about the possible future work in this direction.

2. Review of the MRS router

The MRS router [19] is composed of a recursive coarsening and a recursive refinement process, and features a "V-shaped" work flow, as shown in Figure 1, which is the typical multilevel optimization scheme. The "downward pass" of recursive coarsening builds up the representations of routing regions at different levels, while the "upward pass" of iterative refinement allows a gradual convergence to a globally near-optimal solution.

Before the coarsening process starts, the routing region is divided into an array of tiles of uniform height and width. A 3-D routing graph G_0 is built on top of these tiles, and the routing resource on each edge of the routing graph is calculated. All layout objects such as obstacles, pins and pre-routed wires are counted in the calculation.

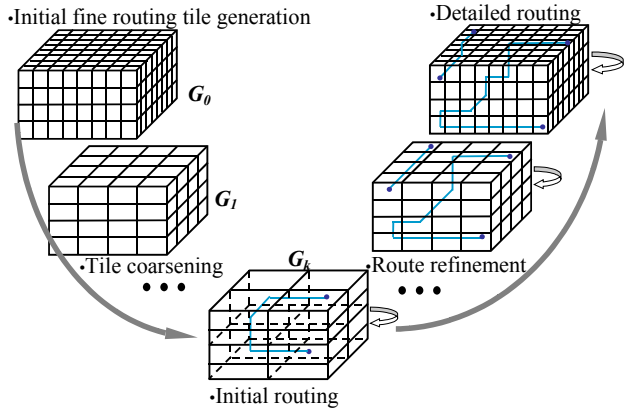


Figure 1 Improved Multilevel Routing Framework

The same resource estimation model as in [12] is used in the MRS router. Three kinds of edge capacities are computed, including the *wiring capacity,* the *interlayer capacity* and the *through capacity.*

The edge capacity C at the left boundary B of the tile in Figure 2 is computed as

$$C = \sum_{i=1}^{4} W_i * D_i / D \qquad (1)$$

where D is the depth of the tile, and the widths and heights of each empty rectangle are denoted as D_i and W_i. To calculate D_i and W_i, we maintain a *contour list* of B, which is defined as a sorted list of the boundaries of all the rectangular obstacles that can be seen from B. In Figure 2, the *contour list* of B is $\{C_1, C_2, C_3, C_4\}$.

The interlayer capacity, which corresponds to the resource that is used by vias, is calculated by the sum of the areas of all empty spaces in the tile. The through capacity, which corresponds to the paths that go straight through a routing tile, is the sum of the boundary capacity contributions of those empty rectangles that span the whole tile. In Figure 2, the through capacity $C_{th} = W_2 + W_4$.

All three kinds of capacities contribute to the path costs. For the example in Figure 3, the total path cost $C_{path} = c_{1,right} + c_{2,left} + c_{2,right} + c_{2,through} + c_{3,left} + c_{3,up} + c_{4,down} + c_{4,right} + c_{5,left}$, where $c_{1,right}$, $c_{2,left}$, $c_{2,right}$, $c_{3,left}$, $c_{4,right}$ and $c_{5,left}$ are the costs related to the wiring capacities of tiles 1, 2, 3, 4 and 5, $c_{2,through}$ is the cost corresponds to the through capacity of tile 2, and $c_{3,up}$, $c_{4,down}$ are the via costs related with the interlayer capacities.

Given the routing graph G_0 at the finest level, the coarsening process generates a series of reduced graphs G_i consecutively, each representing a coarsened level i routing problem P_i with a different resolution.

MRS starts the routing process from the coarsest level k. The corresponding coarsest level routing graph G_k has the smallest number of nodes (usually less than *30×30× number_of_routing_layer*) and the fewest visible nets. A multi-commodity, flow-based routing algorithm, as the one in [8], is used during the *initial routing* at level k to avoid the net ordering problem.

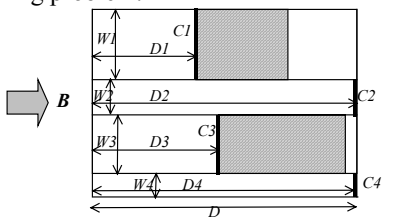


Figure 2 The Boundary Capacity Calculation

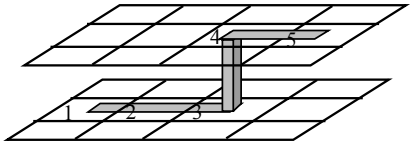


Figure 3 Path Cost Example

With the initial routing result at level k, the refinement process proceeds level-to-level toward the finest level *0*. At each move from a coarser to a finer level, a new sub-routing problem needs to be solved with the guidance from the previous solution. There are two classes of nets to be dealt with at each refinement level. One class is the local nets that just appear at the current level, and the other class is the global nets that have already been planned. Global nets are usually longer than local nets, but there are exceptions due to the dividing line positions, which will be discussed in 3.1. The planned paths at the previous coarser level are projected to the finer routing graph, forming a *preferred region* for each corresponding global net at the current level. There would be an additional penalty assigned to the paths that extend out of the preferred region. For local nets, it would be the first time that they are in the routing process, so they have to be routed from sketch. An A* point-to-point maze-searching algorithm is used in the refinement process at each level. Finally, the implicit connection graph-based gridless routing algorithm in [23] is applied to finalize all nets, each net routed within the corridor of the planning result.

However, the MRS router has several limitations, which prohibit the multilevel routing framework from further improving the routing quality. First, the coarsening process of the MRS is too primitive. When moving from a finer level i to a coarser level $i+1$, the coarsening process simply adds up the capacities of the small tiles and takes the sums as the initial capacities for the edges of G_{i+1}. The problem with this approach is that the resulting routing graph G_i does not consider the routing capacity taken by the nets local to level i. Therefore, during the refinement process, the router would have no idea about the nets local to the level, and may plan many nets in some area which is already congested by many local nets, as shown in Figure 5(a). This problem is more obvious in standard cell circuit designs, since there are more short local nets, as shown in Table 2.

Second, a simple geometric-based minimum spanning tree, which considers only the Manhattan distance but no congestion, is used to decompose the multipin nets in MRS. In modern IC designs, there may be IP blocks or macro cells, which obstruct the routing region with very large obstacles. It is important that the global routing engine would be aware of the large obstacles as well as the congested areas. The congestion-driven planning would require a graph-based tree structure, which is helpful in optimizing both the congestion and the total wire length.

Finally, in MRS, the refinement is processed one net at a time in a fixed order only once at each level. This scheme works well when nets are evenly distributed on each level. However, the distribution of the nets may not always be smooth. In some designs, a huge amount of local nets would suddenly appear in a certain level, making the refinement problem at that level particularly difficult. Also, all nets are routed one-by-one in each level, adversely affecting the net-ordering problem. Furthermore, once an inferior solution is obtained at a coarser level, more refinement effort is needed to correct it at finer levels. Limiting to one round of refinement may not be enough to guarantee satisfactory results.

3. Enhancement Techniques in MARS

The overall multilevel routing scheme of MARS is based on that of the MRS router. However, we have developed several novel techniques to overcome the limitations of MRS. (1) For the coarsening process, we develop a resource reservation technique, and also adopt a more accurate resource model. (2) We develop a way of constructing the congestion-driven Steiner tree structure during the initial routing and the refinement stages to replace the Manhattan-distance based MST. (3) For the uncoarsening process, we adopt a history-based multi-iteration refinement scheme.

3.1 Resource Reservation

During the coarsening process, every move from a finer level i to a coarser level $i+1$ requires merging a certain number (2×2 in our implementation) of adjacent small tiles, which are called the *component tiles*, into a large one. In MRS, the resource estimation of the merged tile is obtained by the simple summation of those of the component tiles. In order to improve the accuracy of the resource estimation, we developed a more sophisticated estimation technique. A new contour list for the merged tile is obtained by merging the contour list of each component tile. The edge capacities of the new tiles are computed by (1), according to the new contour list. Figure 4 illustrates the merging process. T_1, T_2, T_3 and T_4, whose left boundaries are B_1, B_2, B_3 and B_4 respectively, are the four tiles to be merged. The contour lists of B_1, B_2, B_3 and B_4 are retrieved and merged into the contour list of the new edge B. Since the contour lists are sorted, the merging process can be accomplished in $O(n)$ time, where n is the number of line segments in the new contour list. With the contour list of B, it is straightforward to derive the empty rectangles abutting B and then calculate the wiring capacity at B.

However, the estimation computed by the above procedure still cannot precisely model the available routing capacities at the coarser granularity, as when the planning engine moves to a coarser level, a subset of the nets in level i might become completely local to one tile, and thus "invisible" at level $i+1$ and higher. In both hierarchical methods and MRS, no effort was made to model these nets, relying on the assumption that they are relatively short and negligible. However, if the number of such local nets is large, a solution to the coarse level problem may not be aware of locally congested areas, which leads to poor planning result.

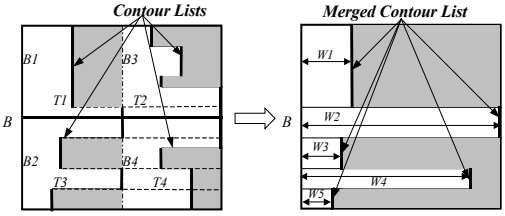


Figure 4 Merging of Contour List

Figure 5(a) shows an example of the effect of the local nets. Net 1 and net 2 are located within a level *1* tile, net 3 is located within a level *2* tile and net 4 spans through several level *2* tiles. Each net is planned without any consideration for the local nets. The paths for net 3 and net 4 will be planned as shown in Figure 5. It is obvious that both path 3 and path 4 may be changed in the finer levels to minimize local congestion. This not only places heavier burden on the refinement procedure, but also wastes the effort spent on the coarser level.

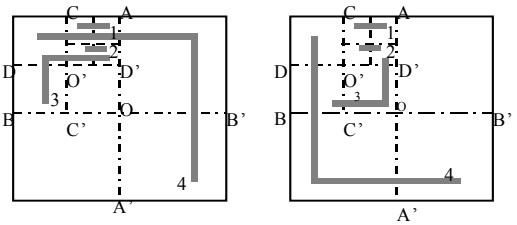
In order to cope with the above problem, we further predict the portion of the resource that would be used by nets that are local to each level, and then reserve the corresponding amount for those nets explicitly in the routing graph. This process is called *resource reservation*.

More specifically, suppose the coarsening process goes through levels *0*, *1*, … *k*, with level 0 being the finest level. Let $c_{i,j}$ denote the initial capacity of edge $e_{i,j}$ in routing graph G_i, and the *capacity vector* $C_i = [c_{i1}\ c_{i2}\ \ldots\ c_{im}]$ represent all routing capacities at level i. Let $T_{n,i}$ = {the set of tiles in which the pins of net n are located on level i}, which is called the *spanning tile set of net n on level i*. The

level of net n, *level*(*n*), is defined as the level above which all pins of *n* are within the boundary of one tile. *level*(*n*) can also be calculated by

$$level(n) = \begin{cases} k & \text{if } |T_{n,k}| > 1 \\ -1 & \text{if } |T_{n,0}| = 0 \\ \max\{i \mid |T_{n,i}| > 1 \text{ and } |T_{n,i+1}| = 1\} & \text{otherwise} \end{cases} \quad (2)$$

Let $L_i = \{n \mid level(n) = i\}$, L_i is called the *local net set* on level *i*. Let $M_i = \{n \mid level(n) > i\}$, M_i is called the *global net set* on level *i*. To better estimate the local nets, we use the maze routing engine to find a path for each net in L_i at level *i*. Then we deduct the routing capacity taken by these local nets in resource reservation.



(a) Effect of Local Nets (without resource reservation)

(b) After Resource Reservation

Figure 5 The Effect of Resource Reservation

Figure 6 shows an example of the reservation calculation at edge CD, AC and BD of a level *i+1* tile. *s* and *t* are pins on a horizontal layer. An L-shaped path connects *s* and *t*. The capacities on CD, AC and BD are first calculated by (1). After the horizontal wire is added, one segment in the contour list of CD will be pushed right by *h*. Therefore, the reserved capacity $r = w * h_1 / h$, where *w* is the wire width. Similarly, the vertical resource reservations on AC and BD are $w * v_1 / v$ and $w * v_2 / v$, a respectively. However, since pins are treated as obstacles in contour list generation, the capacity reservation on AB remains zero. A vector $R_{i+1} = [r_{i+1,1}\ r_{i+1,2}\ \ldots r_{i+1,j}]$ can be obtained by repeating this process, each member corresponding to the reservation on $e_{i+1,j}$ in G_{i+1}. The routing capacity of edges in G_{i+1} is then updated as

$$C'_{i+1} = C_{i+1} - R_{i+1} = [c_{i+1,1} - r_{i+1,1}\ c_{i+1,2} - r_{i+1,2}\ \ldots\ c_{i+1,j} - r_{i+1,j}]$$

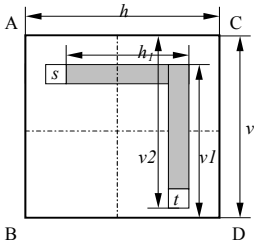


Figure 6 The Reservation Calculation

Figure 5(b) shows the effect of the resource reservation. Net 1 and net 2 are routed at level *0* with reservation made for them on CO' and AD'. On level *1*, net 3 will take a route different from that of Figure 5(a), since there is resource reservation for local nets on level *0*. For the same reason, net 4 is routed on level *2* with a different route.

Since the spanning tiles of each net are at most two tiles away from one another, the maze routing engine will not explore many nodes before it reaches the destination, so the reservation procedure is very fast.

One possible drawback would be that the local nets are unnecessarily treated with higher priorities. However, the routes taken during this phase are usually short and straight, so the reservation amounts are probably only the lower bounds of the resources actually needed by the nets. Furthermore, the reserved routes are not taken as fixed. They can be changed when necessary during the refinement process.

3.2 Congestion-driven Steiner Tree Construction

Instead of using the Manhattan distance-based minimum spanning tree algorithm in MRS, MARS uses a congestion-driven graph based Steiner tree structure to decompose the multipin nets. The Steiner tree heuristic is enabled by a point-to-path maze-searching algorithm both in the initial routing and in the refinement phase. The Steiner tree-based net decomposition results in better wirelength and routability than the MST based decomposition.

Usually, the rectilinear Steiner trees (RST) are used in the decomposition of the multipin nets in the global routing phase. Since the problem of minimum RST has been proved to be NP-hard, many heuristic algorithms (such as [24][25]) were proposed to find the approximate solutions. Most of the Steiner tree approximation algorithms are geometric distance based. However, it is important that the global routing engine be aware of the large obstacles as well as the congested areas. Congestion-driven planning requires computing Steiner tree on a routing graph, which encodes the routing capacity and length information, so that both the congestion and wirelength are optimized. This is much more difficult than computing a simple Manhattan distance-based tree, as the commonly used graph-based Steiner heuristic requires computing all-pair shortest paths which has the time complexity of $O(n^3)$.

In [20][21], Steiner tree approximation algorithms that consider congestion were proposed, yet the topologies generated are limited to the initial geometric distance-based spanning tree structure and may not work well for circuits with large obstacles. A graph-based A-tree algorithm is proposed in [22], which could avoid large obstacles. The runtime is also reasonable, since all-pair shortest paths are not necessary for the construction of an A-tree. However, an A-tree is limited to optimize the paths from the source to all targets and the total wire length of an A-tree depends on the position of the source. Therefore, the graph-based A-tree topology may not be suitable for the decomposition of the non-critical nets.

In our routing system, the congestion-driven Steiner tree is constructed during the initial routing and refined in an incremental way starting from *level(n)* to level 0. The maze search engine is used to find the congestion-driven Steiner tree edges.

The tree for net *n* is initially constructed at *level(n)*, where *n* first spans more than one tile. Let $P_{n,i}$ denote the

set of nodes in G_i corresponding to the tiles that the pins of *n* are located at level *i*. The first step is to find a geometric-based MST T_{MST} for $P_{n,i}$ in G_i. Then the edges of T_{MST} are sorted by their geometric distances. For each of the T_{MST} edge, we use a maze search algorithm to find a shortest path on the routing graph. Instead of the point-to-point routing, the search process would stop whenever any existing paths connecting to the target point are visited and the hit point becomes a Steiner point. If there are multiple minimum paths, we choose the one that is closest to the center of all pins of the net. The Steiner tree is composed of all paths of the edges of T_{MST} .

After initial tree construction at *level(n)*, we further continue the construction by connecting the newly appeared nodes of the net to the tree through the modified point-to-path maze searching algorithm at each refinement level.

In the spanning tree decomposition method, the exact locations of the two end points of each edge are fixed. Therefore, the decomposition of the multi-pin nets into several two-pin nets is straightforward. The resulting two-pin nets, which correspond to the MST edges, are independent of each other. However, in our Steiner tree decomposition method, the Steiner point locations are floating, which means we have edges (two-pin nets) whose one end is not a pin, but another edge (two-pin net). Therefore, that edge is constrained by its target edge of the same tree. To solve this problem, within each multi-pin net, we keep an ordering of all Steiner tree edges (two-pin nets) based on the routing order of their first route. During the refinement, the newly appeared nodes are connected first, and the new edges are inserted to the head of the ordering. Then the global nets are refined according to the ordering we get from the previous levels. Using this method, we can gradually construct a Steiner tree without a priori fixing the Steiner points.

Figure 7 shows an example of the formation of a Steiner tree from level 2 to level 0. The label beside each edge shows the ordering of that edge. At level 2, there are three pins, a, b and c, the Steiner tree is ab(1), cb(2) (the numbers in the parentheses are the orderings of the edges). At level *1*, two new pins, d and e appear. We first connect the two new pins to the tree, resulting ae(1) and db(2), then we refine the global nets, be(3), cT'(4), where T' is the Steiner point. At level *0*, no new pins appear, and the nets are refined by the ordering, ae(1), db(2), eT''(3), cT'(4), where T'' is the new Steiner point is added in the tree.

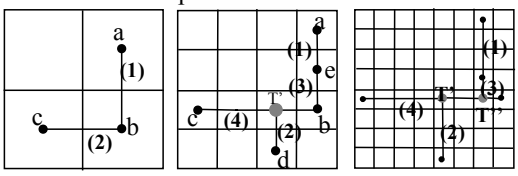


Figure 7 The Gradual Construction of a Steiner Tree

3.3 Multi-iteration Refinement

MARS uses a history-based iterative refinement similar to the one in [6] at each level of refinement. The main idea of the history-based method is to iteratively update the edge routing cost with the consideration of historical congestion information and re-route all nets based on the new edge cost functions. The cost of edge *e* during the *ith* iteration is calculated by:

$$Cost(e,i) = \alpha * congestion(e,i) + \beta * history(e,i) \quad (3)$$

$$history(e,i) = history(e,i-1) + \gamma * congestion(e,i-1) \quad (4)$$

where *congestion(e,i)* is a three-tier slope function of the congestion on *e, history(e,i)* is the history cost, indicating how congested that edge was during the previous iterations, and α, β, γ are scaling parameters.

The congestions of the routing edges are updated every time a path of a net is routed. Since routing tile at coarser levels can be quite large compared to a normal global routing tile size, special attention needs to be paid to resource updating. The common way is to map all pins to the center of a routing tile, and deduct the same routing resource for all paths passing the same routing graph edge. When the routing tile is large as the case in the top levels in multilevel routing approach, this kind of rough abstraction may introduce too much error. Let us look at the example in Figure 8. The routing region is divided into 16×16 tiles at level *0*. Suppose there are 4 levels for the design, so at the coarsest level *3*, the routing region is divided into 2×2 tiles. Though N1 is short, it may still be visible at level (level *3*) because of the diving line position. When updating the resource after routing *N1* at the initial routing phase, *1/16* of the total resource will be subtracted from both *e1* and *e2* by the rough method according to our resource model. But the actual resource deduction, by considering the real pin locations, would only be *1/64* of the total edge capacity. In MARS, the precise real pin locations are used in resource updating to avoid such inaccuracy.

After each iteration, the history cost of each edge is increased according to (4). Then the congestions of all edges are scanned to determine whether another iteration is necessary. If so, all edge usage are reset to zero and the refinement process at the same level is restarted.

Multiple iterations at every level may be time consuming when the routing graph is large. We try to control the planning runtime by the level number. We also make the planning engine iterate more rounds at the coarser levels than at the finer levels to improve the quality and the run-time.

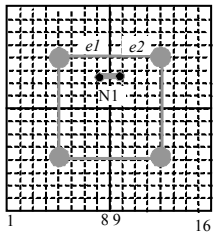


Figure 8 The Accurate Resource Deduction

3.4 Other Enhancements

As in MRS, MARS uses the implicit connection graph-based gridless routing algorithm in [23] to finalize and evaluate the multilevel planning result. However, we made two changes to the detailed routing algorithm to better

cooperate with the current multi-level routing scheme. First, we added a local rip-up and reroute routine to handle the failed nets. This local modification can alleviate part of the local net ordering problem, so that the final routing results will better correlate with the planning quality. Second, to support the Steiner tree construction, we modified the detailed router from a pin-to-pin router to a pin-to-pin or pin-to-path router. This method provides a simple yet effective method to find the best location of the Steiner points, at slight increase of runtime due to additional target-hit check after each expansion.

4. Experimental Results

We implemented our routing system, MARS, on a Sun Blade 750 using C++. This system has been tested on a wide range of test cases. Table 1 lists the detailed information of our testing examples. Among them, s5378 to s38584, and struct are standard cell circuits, Mcc1, Mcc2 and Raytheon are MCM circuits. The column labeled "#Div." shows the finest tiles number at the finest level of each example. Our comparisons of different techniques are mainly based on the number of nets that fail to complete after the detail routing phase.

Table 2 shows the net distribution on each planning level. For standard cell circuits, there is an average of 35% nets residing in the finest level, or even lower (i.e. can not be seen at the finest level). Starting from the finest level, the number of global nets decreases at a steady pace when going from the finer to the coarser levels. The distribution of nets on each level is quite smooth. However, for the MCM test cases, most of the nets are located on the coarsest level. Having a clear picture of the net distribution on each level is helpful in understanding the effects of the techniques discussed in this paper.

4.1 Impact of Resource Reservation

We first evaluate our resource reservation algorithm during the coarsening phase. Table 3 compares the results that are generated with and without resource reservation. It shows that the resource reservation technique is effective for the standard cell cases. It helps to decrease the number of failed nets for all standard cell cases with less than 20% increase of runtime. This is because that standard cell cases have a rather smooth distribution of connections, and that resource reservation helps to get a more global picture of the whole routing problem at the coarser levels. However, when applied to MCM cases, resource reservation does not show improvement. The reason is that the number of local nets on finer levels is very small, so making reservation for them does not help much.

4.2 Impact of the Use of Congestion-driven Steiner Tree

We then compare the effect of different tree structures in multipin net decomposition. We implemented three different types of tree structures. All three methods start from a simple Manhattan distance-based MST. The geometric based-MST (S-MST) method only uses the Manhattan distance-based minimum spanning tree topologies as the candidates for the coarsest level flow-based algorithm. The graph-based MST (G-MST) version then uses a maze-searching algorithm to search for paths at top level to minimize congestion. The graph-based Steiner tree (G-ST) is generated by the method in Section 3.2.

Table 4 shows the results of the three different tree decompositions. In most cases, the use of a graph-based tree leads to better completion rate. Also, G-ST further reduces the wirelength and improves the completion rate. The G-MST and G-ST algorithms are more complicated, and the tradeoff for the improvement might be runtime. However, in our experiments, G-MST and G-ST are actually faster than S-MST as shown in Table 4. The reason is that the uses of G-MST and G-ST result to better planning results, so the detailed routing engine spends less time to find the paths for all nets.

4.3 Impact of the Use of Multi-iteration

We try MARS with and without multi-iterations and compare the results in Table 5. For standard cell circuits, the improvement due to multi-iterations is less than that by the resource reservation method. However, for the MCM cases, where the resource reservation method is not so effective, multi-iteration can still reduce the number of failed nets after detail routing. This is because multiple iterations on the same planning level would reduce the influence that routing order has on solution quality. Again, since we are running more times of maze searching during each refinement level, the runtime would grow with the size of chip area.

4.4 Overall Improvement

We integrate all the above enhancement techniques discussed in Section 3 into the MARS router. The results are compared with that of MRS [19]. We run the test cases both with and without ripup and reroute at the detailed routing phase. As we have discussed before, the ripup and reroute procedure can alleviate the local net ordering problem during the detailed routing phase, and make the final results more related with the planning quality. However, since we use the Steiner tree structure in MARS, in order to make the problem easier, we fixed some pre-routed nets that contain a Steiner point of other nets. This approach to some extent restricts the searching range of the ripup and reroute process.

Table 6 shows the results without ripup and reroute, and Table 7 shows the results with ripup and reroute. The MARS router can decrease the failed nets by 2× to 18× with roughly 50% increase of runtime. Our techniques are also effective in reducing wirelength and via number. It is understandable that the overall improvement can not be as much as the sum of the individual technique improvements, since the different techniques would have similar effects.

5. Conclusions

We proposed several new techniques to improve the multilevel routing system, and developed the enhanced router, MARS. The resource reservation and exact resource calculation allow the coarsening process to generate a set of

smaller sub routing problems that accurately reflect the original routing problem. We also developed a heuristic to gradually generate a graph-based Steiner tree, which can help to reduce both the wirelength and the congestion. For the refinement process, we introduced a history-based multi-iteration algorithm to further optimize the final results. These techniques make the multilevel routing system more complete and powerful.

There are several research directions for further enhancements. One possible improvement is a more clever and systematic way of routing region division and routing graph generation. Both the routing tile size and the number of tiles to be merged during the coarsening process can vary according to the actual circuit design. We tried some ideas of non-uniform merging during the coarsening process. Though we have not achieved consistently better results yet, we still believe that non-uniform coarsening may be a good way to reasonably distribute the computing resource to different regions of the layout according to the actual needs. Also, since the multilevel framework has been successfully applied to partitioning, placement, and routing, it is interesting to investigate if there exists a unified way of integrating these algorithms into a single powerful multilevel optimization flow for VLSI physical design.

Reference

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

[2] J. Soukup, "Fast maze router," Proc. 15th Design Automation Conference, pp. 100-102, 1978

[3] K. Mikami and K. Tabuchi, "A computer program for optimal routing of printed circuit connectors," IFIPs Proc, vol. H-47, pp.1475-1478, 1968.

[4] D. Hightower, "A solution to line routing problems on the continuous plane," Proc. IEEE 6th Design Automation workshop, pp. 1-24, 1969.

[5] R. Nair., "A simple yet effective technique for global wiring, " IEEE Trans. on Computer-Aided Design, CAD-6(2), 1987.

[6] L. McMurchie and C.Ebeling, "Pathfinder: a negotiation-based performance-driven router for FPGAs", Proc. of 3rd International ACM/SIGDA Symposium on Field-Programmable Gate Arrays, Feb. 1995, pp.111-117.

[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," 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," Proc. 35th Design Automation Conference, pp. 356-361, Jun.1998

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

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

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

[13] M. Burstein and R. Pelavin, "Hierarchical channel router," Proc. of 20th Design Automation Conference, pages 519-597, 1983.

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

[15] D. wang and E. Kuh, "A new timing driven multiplayer MCM/IC routing algorithm," Proc. IEEE Multi-Chip module Conference, pp. 89-94, Feb. 1997.

[16] 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

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

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

[19] J. Cong, J. Fang and Y. Zhang, "Multilevel Approach to Full-Chip Gridless Routing," Proc. IEEE International Conference on Computer Aided Design, San Jose, California, pp. 396-403, Nov. 2001

[20] C. Chiang, M. Sarrafzadeh, and C.K, Wong, "A powerful global router: Based on Steiner min-max trees," Proc. IEEE International Conference on Computer-Aided Design, pp. 2-5, Nov. 1989.

[21] C. Chiang, M. Sarrafzadeh, and C.K., Wong, "A weighted-Steiner-tree-based global router," Manuscript, 1992.

[22]. J. Cong, A. B. Kahng and K.-S. Leung, "Efficient algorithms for the minimum shortest path Steiner arborescence problem with applications to VLSI physical design," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol.17, no. 1, pp. 24-39, Jan. 1999

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

[24].M. Borah, R. M. Owens, M. J., Irwin, **"**An edge-based heuristic for Steiner routing**,"** IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol.13, (no.12), Dec. 1994. p.1563-8

[25] J. Griffith, G. Robins, J. S. Salowe, Tongtong Zhang, **"**Closing the gap: near-optimal Steiner trees in polynomial time," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol.13, (no.11), Nov. 1994. p.1351-65

Circuits	#Nets	#2 pin nets	# Lay-ers	#Lev-els	#Div.
S5378	1694	3124	3	3	61x34
S9234	1486	2774	3	3	57x32
S13207	3781	6995	3	4	92x51
S15850	4472	8321	3	4	98x55
S38417	11309	21035	3	3	159x86
S38584	14753	28177	3	4	180x94
Struct	1920	3551	3	5	273x273
Primary1	904	2037	3	3	53x35
Primary2	3029	8197	3	6	73x46
Mcc1	802	1694	4	2	25x22
Mcc2	7118	7541	4	3	43x43
Raytheon	249	430	4	4	28x16

Table 1 Test Circuit

Circuits	<level 0	Level 0	Level 1	Level 2	Level 3	Level 4
S5378	593	374	648	1509	0	0
S9234	624	362	591	1197	0	0
S13207	1522	880	1445	1187	1961	0
S15850	1814	1086	1835	1434	2152	0
S38417	4732	2940	4220	9143	0	0
S38584	6103	3821	6362	4833	7058	0
struct	37	196	364	494	671	1752
Primary1	1	466	465	1105	0	0
Primary2	15	648	163	7371	0	0
Mcc1	97	221	1376	0	0	0
Mcc2	0	663	163	7371	0	0
Raytheon	12	44	113	53	108	108

Table 2 Net Distributions at Each Level

Circuits	With no RR		With RR	
	#Failed nets	Runtime (s)	#Faied nets	Runtime (s)
S9234	38	18.4	23	19.8
S13207	131	70.3	89	79.8
S15850	144	98.8	92	116.0
S38417	371	349.3	312	401.8
S38584	490	1012.9	331	1090.6
Struct	26	321.5	12	297.6
Primary1	16	19.9	16	20.4
Primary2	23	154.3	23	152.7
Mcc1	55	140.0	68	212.3
Mcc2	148	2828.0	164	2738
Raytheon	29	18.4	17	17.5
Avg.	1.36	1	1	1.09

Table 3 Impact of Resource Reservation

Circuits	S-MST		G-MST		G-ST	
	#F. net	Run time (s)	#F. net	Runtime (s)	#F. net	Run-time (s)
S5378	108	23.7	81	20.72	65	24.8
S9234	81	16.4	53	15.44	38	18.4
S13207	334	58.5	143	55.36	131	70.3
S15850	301	89.6	163	80.42	144	98.8
S38417	879	374.2	467	303.16	371	349.3
S38584	1294	583.0	573	946.12	490	1012.9
Struct	29	148.5	31	204.6	26	321.5
Primary1	1	137	20	20.7	16	19.9
Primary2	4	541	60	163	23	154.3
Mcc1	43	412.9	65	109.7	55	140.0
Mcc2	982	13562	197	2680.4	148	2828.0
Raytheon	21	16.1	35	17.3	29	18.4
Avg.	1.91	1	1.34	0.79	1	0.95

Table 4 Impact of Different Tree Structures

Circuits	No iteration		With multi-iteration	
	#Failed Nets	Runtime (s)	#Failed Nets	Runtime (s)
S5378	65	24.8	37	30.9
S9234	38	18.4	32	21.6
S13207	131	70.3	93	73.3
S15850	144	98.8	71	138.4
S38417	371	349.3	256	481.1
S38584	490	1012.9	341	1218.7
Struct	26	321.5	26	576.8
Primary1	16	19.9	16	22.2
Primary2	23	154.3	23	165.4
Mcc1	55	140.0	17	165.0
Mcc2	148	2828.0	92	3611.5
Raytheon	29	18.4	15	15.8
Avg.	1.56	1	1	1.23

Table 5 Impact of Multi-iterations in each Refinement

Circuits	MRS				MARS			
	#Failed Nets	Runtime (s)	Wire length	#Vias	#Failed Nets	Runtime (s)	Wire length	#Vias
S5378	108	23.7	8.8e7	7116	44	34.3	7.9e7	7112
S9234	81	16.4	6.7e7	6012	13	24.4	5.9e7	6133
S13207	334	58.5	2.5e8	15160	66	115.4	1.9e8	15691
S15850	301	89.6	2.9e8	18206	70	152.1	2.3e8	18644
S38417	879	374.2	7.7e8	44614	274	567.6	5.1e8	45107
S38584	1294	583.0	1.2e9	59916	338	1327.4	6.9e8	56437
Struct	29	148.5	9.4e8	9540	12	525.4	8.5e8	8355
Primary1	1	136.9	1.0e9	6394	16	22.6	1.0e9	5447
Primary2	4	541.1	4.2e9	26900	23	170.8	4.2e9	23071
Mcc1	43	412.9	3.0e10	5127	27	153.7	2.7e10	4848
Mcc2	982	13562.3	5.2e11	26579	108	3801.9	4.0e11	34191
Raytheon	21	16.1	2.4e8	954	18	21.1	2.1e8	1109
Avg.	3.3	1.67	1.23	1.0	1	1	1	1

Table 6 The Results Compared with MRS (no ripup and reroute)

Circuits	MRS				MARS			
	#Failed Nets	Runtime (s)	Wire length	#Vias	#Failed Nets	Runtime (s)	Wire length	#Vias
S5378	21	31.7	7.9e7	7408	8	34.3	8.0e7	7197
S9234	14	20.4	5.9e7	6238	3	24.4	5.9e7	6155
S13207	80	85.9	2.0e8	15958	12	115.4	1.9e8	15832
S15850	89	171.2	2.4e8	18960	10	154.6	2.3e8	18778
S38417	177	459.7	5.5e8	46614	42	567.6	5.0e8	45620
S38584	304	791.9	8.1e8	62862	46	1308.2	7.0e8	63205
Struct	4	154.7	8.8e8	9616	2	529.0	8.5e8	8353
Primary1	1	135.8	1.0e9	6394	1	22.6	1.0e9	5481
Primary2	0	533.5	4.2e9	26910	2	173.5	4.2e9	23037
Mcc1	17	521.3	3.0e10	5147	17	182.5	2.7e10	4874
Mcc2	826	28833	5.2e11	27444	46	4367.4	4.1e11	34463
Raytheon	16	28.0	2.3e8	976	13	28.3	2.2e08	1128
Avg.	4.74	2.07	1.07	1.02	1	1	1	1

Table 7 The Results Compared with MRS (with ripup and reroute)