A Novel Framework for Multilevel Routing Considering Routability and Performance *

Shih-Ping Lin[1] *and Yao-Wen Chang*[2]

[1]Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan

[2]Graduate Institute of Electronics Engineering & Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

Abstract

We propose in this paper a novel framework for multilevel routing considering both routability and performance. The two-stage multilevel framework consists of coarsening followed by uncoarsening. Unlike the previous multilevel routing, we integrate global routing, detailed routing, and resource estimation together at each level of the framework, leading to more accurate routing resource estimation during coarsening and thus facilitating the solution refinement during uncoarsening. Further, the exact routing information obtained at each level makes our framework more flexible in dealing with various routing objectives (such as crosstalk, power, etc). Experimental results show that our approach obtains significantly better routing solutions than previous works. For example, for a set of 11 commonly used benchmark circuits, our approach achieves 100% routing completion for all circuits while the previous multilevel routing, the three-level routing, and the hierarchical routing can complete routing for only 3, 0, 3 circuits, respectively. In particular, the number of routing layers used by our router is even smaller. We also have performed experiments on timing-driven routing. The results are also very promising.

1 Introduction

Research in VLSI routing has received much attention in the literature. Routing is typically a very complex combinatorial problem. In order to make it manageable, the routing problem is usually solved using the two-stage approach of global routing followed by detailed routing. Global routing first partitions the routing area into tiles and decides tile-to-tile paths for all nets while detailed routing assigns actual tracks and vias for nets. Many routing algorithms adopt a flat framework of finding paths for all nets. Those algorithms can be classified into sequential and concurrent approaches. Early sequential routing algorithms include maze-searching approaches [16, 21] and line-searching approaches [13], which route net-by-net. Most concurrent algorithms apply network-flow or linear-assignment formulation [1, 20] to route a set of nets at one time.

The major problem of the flat frameworks lies in their scalability for handling larger designs. As technology advances, technology nodes are getting smaller and circuit sizes are getting larger. To cope with the increasing complexity, researchers proposed to use hierarchical approaches to handle the problem: Marek-Sadowska proposed a hierarchical global router based on linear assignment [19]; Heisterman and Lengauer presented a hierarchical integer linear programming approach for global routing [12]; Wang and Kuh proposed a hierarchical (α, β)* algorithm for timing-driven multilayer MCM/IC routing [22]; Chang, Zhu, and Wong applied linear assignment to develop a hierarchical, concurrent global and detailed router for FPGA's [3].

The two-level, hierarchical routing framework, however, is still limited in handling the dramatically growing complexity in current and future IC designs which may contain hundreds of millions of gates in a single chip. As pointed out in [5], for a 0.07 μm process technology, a 2.5 × 2.5 cm^2 chip may contain over 360,000 horizontal and vertical routing tracks. To handle such high design complexity, the two-level, hierarchical approach becomes insufficient. Therefore, it is desired to employ more levels of routing for larger IC designs.

The multilevel framework has attracted much attention in the literature recently. It employs a two-stage technique: coarsening followed by uncoarsening. The coarsening stage iteratively groups a set of circuit components (e.g., circuit nodes, cells, modules, routing tiles, etc) based on a predefined cost metric until the number of components being considered is smaller than a threshold. Then, the uncoarsening stage iteratively ungroups a set of previously clustered circuit components and refines the solution by using a combinatorial optimization technique (e.g., simulated annealing, local refinement, etc). The multilevel framework has been successfully applied to VLSI physical design. For example, the famous multilevel partitioners, *ML* [2], *hMETIS* [14], and *HPM* [8], the multilevel placer, *mPL* [4], and the multilevel floorplanner/placer, *MB*-tree* [18], all show the promise of the multilevel framework for large-scale circuit partitioning, placement, and floorplanning.

Recently, Cong, Fang, and Zhang proposed a pioneering multilevel approach for large-scale, full-chip, routability-driven routing [5]. The framework starts by recursively coarsening routing tiles, and an estimation of routing resources is computed at each level. When the number of tiles is below a threshold, a multicommodity flow algorithm is used to obtain an initial routing solution. Then the uncoarsening stage uses a modified maze-searching algorithm to further improve the routing solution level by level. Their final results of the multilevel algorithm are tile-to-tile paths for all the nets. The results are then fed into a detailed router to find the exact connection for each net. Their experimental results show better routing quality or running times than the traditional two-stage flat approach of global routing followed by detailed routing and the hierarchical approaches.

Inspired by the work of the multilevel router presented in [5], we propose in this paper a novel framework for multilevel routing considering *both routability and performance*. Different from the work presented in [5], ours has the following distinguished features:

- We integrate global routing, detailed routing, and resource estimation together at each level of the framework, leading to more accurate routing resource estimation during coarsening and thus facilitating the solution refinement during uncoarsening. Specifically, at each level of the coarsening stage, we perform global routing to obtain a good initial solution for all nets inside the tiles being considered and then detailed routing to obtain the exact routing patterns for these nets. Since the exact routing patterns are known, resource estimation is more accurate. With these good properties, the refinement conducted at the uncoarsening stage becomes much easier. In contrast, the work [5] performs only resource estimation during the coarsening stage and only global routing during the uncoarsening stage. After the multilevel processing is finished, the final global routing result is then fed into

*This work was partially supported by the National Science Council of Taiwan ROC under Grant No. NSC 91-2215-E-002-038. E-mails: is85060@cis.nctu.edu.tw; ywchang@cc.ee.ntu.edu.tw.

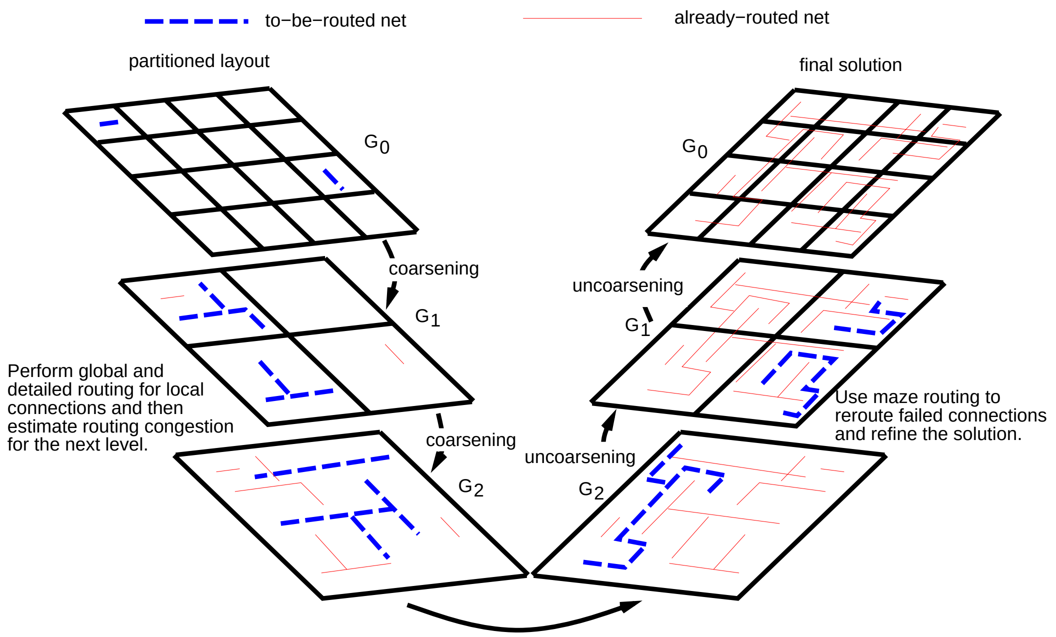
to−be−routed net
already−routed net
partitioned layout
final solution
G0
G0
coarsening
uncoarsening
G1
G1
Perform global and detailed routing for local connections and then estimate routing congestion for the next level.
coarsening
uncoarsening
G2
G2
Use maze routing to reroute failed connections and refine the solution.

Figure 1: The multilevel framework flow.

a detailed router to obtain the final routing solution. It is obvious that our approach can have better interaction among global routing, detailed routing, and resource estimation since they are considered simultaneously. For example, global and detailed routers usually use rip-up and re-route to refine a routing solution based on the results of resource estimation. If the three tasks are performed separately, the re-routing process conducted at the global routing stage may be in vain since it does not know if the re-routing is useful for the detailed router. Also, the detailed router may fail to find a path because of the low flexibility induced from the separated global routing. Therefore, making the three tasks interact with each other can significantly improve routing quality.

- A two-stage refinement method of Z-pattern routing followed by maze routing is used in our multilevel framework, which makes re-routing much more effective.
- Unlike the work [5] that considers routability alone, we also apply a recalling modification method to perform timing-driven routing.
- Our framework is more flexible and thus different routing objectives (such as crosstalk, power, etc) can be incorporated into our framework since exact track and wiring information at each level after detailed routing is known.

Figure 1 shows our multilevel framework, and Table 1 summaries the differences between our multilevel system and that presented in [5].

Experimental results show that our approach obtains significantly better routing solutions than the multilevel routing [5], the three-level routing [6], and the hierarchical approach [5]. For the 11 benchmark circuits provided by the authors of [5], our approach obtains 100% routing completion for all circuits while the multilevel routing, the three-level routing, and the hierarchical routing can complete routing for only 3, 0, 3 circuits, respectively. In particular, the number of routing layers required for routing completion for our approach is even smaller. We also have performed experiments on timing-driven routing. The results are also very promising.

The rest of this paper is organized as follows. Section 2 presents the routing model and the multilevel routing framework. Section 3 presents our framework for routability and timing optimization. Experimental results are shown in Section 4. Finally, we give concluding remarks in Section 5.

2 Preliminaries

2.1 Routing Model

Routing in modern IC's is a very complex processing, and thus we can hardly obtain solutions directly. Our routing algorithm is based on a graph search technique guided by the congestion and timing information associated with routing regions and topologies. The router assigns higher costs to route nets through congested areas to balance the net distribution among routing regions. For performance-driven routing, additional costs are added to the routing topologies with longer critical path delays.

Before we can apply the graph search technique to multilevel routing, we first need to model the routing resource as a graph such that the graph topology can represent the chip structure. Figure 2 illustrates the graph modeling. For the modeling, we first partition a chip into tiles. A node in the graph represents a tile in the chip, and an edge denotes the boundary between two adjacent tiles. Each edge is assigned a capacity according to the physical area or the number of tracks of a tile. The graph is used to represents the routing area and is called *multilevel routing graph* G_0. A global router finds tile-to-tile paths for all nets on G_0 to guide the detailed router. The goal of global routing is to route as many nets as possible while meeting the capacity constraint of each edge and any other constraint, if specified. As the process technology advances, multiple routing layers are possible. The number of layers in a modern chip can be more than six [11]. Wires in each layer run either horizontally or vertically. We refer to the layer as a horizontal (H) or a vertical (V) routing layer.

2.2 Multilevel Routing Model

As illustrated in Figure 1, G_0 corresponds to the routing graph of the level 0 of the multilevel coarsening stage. At each level, our global

	Our framework	The framework in [5]
Objective	• Considers both routability and timing.	• Considers only routability.
Coarsening	• Performs global and detailed routing at each level. • Performs congestion estimation after detailed routing. • Uses the Z-shaped routing refinement method.	• Performs only routing resource estimation using a line-sweep algorithm.
After coarsening	• No initial routing is needed.	• Initial routing using a multicommodity flow algorithm.
Uncoarsening	• Uses a global and a detailed maze routers to refine the solution.	• Uses a global maze router to refine the solution.
Characteristics	• Performs routing during coarsening and thus detailed routing information and local congestion are known. • Activates refinement (rip-up and re-route) when detailed routing fails and is thus more effective for actual routing. • Performs global and detailed routing at each level.	• Coarsening does not route any net, lacking local routing information. • Activates refinement (rip-up and re-route) during global routing and may not be useful to detailed routing. • Performs global and detailed routing separately.

Table 1: Framework comparison between ours and [5].

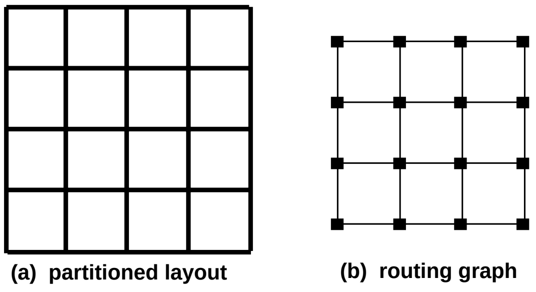

Figure 2: The routing graph.

router first finds routing paths for the *local nets* (or *local 2-pin connections*) (those nets [connections] that entirely sit inside a tile), and then the detailed router is used to determine the exact wiring. After the global and detailed routing are performed, we merge four adjacent tiles of G_0 into a larger tile and at the same time perform resource estimation for use at the next level (i.e., level 1 here). Coarsening continues until the number of tiles at a level, say the k-th level, is below a threshold. After finishing coarsening, the uncoarsening stage tries to refine the routing solution starting from the last level k where coarsening stops. During uncoarsening, the unroutable nets during coarsening are considered, and maze routing and rip-up and re-route are performed to refine the routing solution. Then we proceed to the next level (level $k-1$) of uncoarsening by expanding each tiles to four finer tiles. The process continues up to level 0 when the final routing solution is obtained.

3 Multilevel Routing Framework

Our multilevel routing algorithm is inspired by the work [5]. Nevertheless, our framework is significantly different from [5]. During the coarsening stage of the work [5], instead of routing or planning wires, they only estimate routing resources by using a line-sweep algorithm and then recursively coarsen to the last level k. Since their coarsening stage does not perform real routing, it is hard to retrieve the routing information at the higher level, which may make real routing resource estimation inaccurate. At the last level k, they apply a multicommodity flow algorithm to obtain an initial routing and avoid the net ordering problem. However, a router may encounter higher congestion when uncoarsening expands local nets. A bad initial routing at the higher level needs more time to re-route at the lower level because of lacking local routing information. This problem is also with the hierarchical approach.

Our router tends to route shorter nets first since we route local nets at each level of coarsening. It is obvious that the local nets at the lower level (say, level 0) are usually shorter than those at a higher level (say, level k). Naturally, a shorter net enjoys less freedom while searching for a path to route it. This fact holds even during rip-up and re-route. Thus, this observation implicitly suggests that a shorter net has a higher priority than a longer net as far as routability is concerned. Kastner, Bozorgzadeh, and Sarrafzadeh in [15] also suggest this conclusion. Thought this net ordering scheme may not be the optimal solution for some routing problems (for example, when timing is considered, routing the most critical net first often leads to better timing performance), it is still a reasonable alternative.

3.1 Multilevel Routing for Routability

Given a netlist, we first run the minimum spanning tree (MST) algorithm to construct the topology for each net, and then decompose each net into 2-pin connections, with each connection corresponding to an edge of the minimum spanning tree. Our multilevel framework starts from coarsening the finest tiles of level 0. At each level, tiles are processed one by one, and only local nets (connections) are routed. At each level, the two-stage routing approach of global routing followed by detailed routing is applied. (See Figures 3(a)–(c) for an illustration.) The global routing is based on the approach used in the Pattern Router [15] and first routes local nets (connections) on the tiles of level 0. Let the multilevel routing graph of level i be $G_i = (V_i, E_i)$. Let $R_e = \{ e \in E_i \mid e \text{ is the edge chosen for routing}\}$. We apply the cost function $\alpha : E_i \rightarrow \Re$ to guide the routing:

$$\alpha(R_e) = \sum_{e \in R_e} c_e, \tag{1}$$

where c_e is the congestion of edge e and is defined by

$$c_e = 1/2^{(p_e - d_e)},$$

where p_e and d_e are the capacity and density associated with e, respectively.

After the global routing is completed, we perform detailed routing with the guidance of the global-routing results and find a real path in the chip. Our detailed router is based on the maze-searching algorithm and supports the *local refinement* illustrated in Figures 3(d)–(f). Pattern routing uses an L-shaped or a Z-shaped route to make the connection, which gives the shortest path length between two points. Therefore, the wire length is minimum, and thus we do not include wire length in the cost function at this stage. We measure the routing congestion based on the commonly used channel density. After the detailed routing finishes routing a net, the channel density associated with an edge of a multilevel graph is updated accordingly. This is called *resource estimation*.

Our global router first tries L-shaped pattern routing. If the routing fails, we try Z-shaped pattern routing. This can be considered as a simple version of rip-up and re-route. If both pattern routes fail, we give up routing the connection, and an overflow occurs. We refer to a *failed net* (*failed connection*) as that causes an overflow. The failed nets (connections) will be reconsidered (refined) at the uncoarsening stage. There are at least two advantages by using this approach. First, routing resource estimation is more accurate than that performing global routing alone since we can precisely evaluate the routing region. Second, we can obtain a good initial solution for the following refinement very effectively since pattern routing enjoys very low time complexity and uses fewer routing resources due to its simple L-shaped and Z-shaped routing patterns. Figure 3 shows an example of routing a local net in a tile.

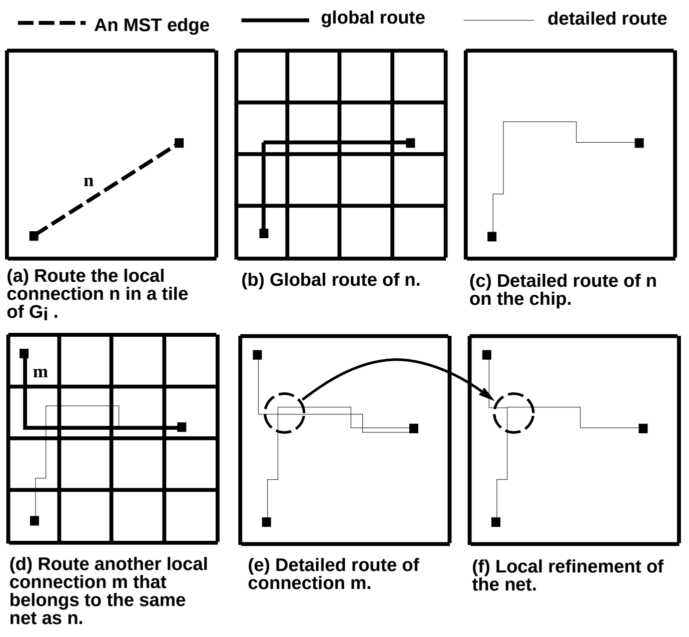
An MST edge
global route
detailed route
n
(a) Route the local connection n in a tile of Gi .
(b) Global route of n.
(c) Detailed route of n on the chip.
m
(d) Route another local connection m that belongs to the same net as n.
(e) Detailed route of connection m.
(f) Local refinement of the net.

Figure 3: Global routing, detailed routing, and local refinement.

The uncoarsening stage starts to refine each local failed net (connection), left from the coarsening stage. The global router is now changed to the maze router with the following cost function $\beta : E_i \rightarrow \Re$:

$$\beta(R_e) = \sum_{e \in R_e} (al_e + bc_e + co_e), \quad (2)$$

where a, b, and c are user-defined parameters, l_e is the length of the net (connection), and $o_e \in \{0, 1\}$. If an overflow happens, o_e is set to 1; it is set to 0, otherwise.

There is a trade-off among minimizing wire length, congestion, and overflow. At the uncoarsening stage, we intend to resolve the overflow in a tile. Therefore, we let c be much larger than a or b. Also, a detailed maze routing is performed after the global maze routing. Iterative refinement of a failed net is stopped when a route is found or several tries (say, three) have been made. Uncoarsening continues until the first level G_0 is reached and the final solution is found. Note that the global maze routing here serves as an elaborate rip-up and re-route processor, in contrast to the simple L-shaped and Z-shaped routing during coarsening. This two-stage approach of global and local refinement of detailed routing gives our overall refinement scheme.

3.2 Multilevel Routing for Performance

3.2.1 Timing Optimization

In deep submicron IC designs, interconnection delay dominates the performance of a circuit. Therefore, improving the wire delay also improves the overall chip performance. The routing problem with timing constraints is much more complex, as not only congestion must be controlled but also timing constraints must be satisfied. Many techniques have been developed to facilitate high-performance IC designs. For example, the algorithms for performance-driven routing-tree topology construction have received much attention [7, 10, 17]. However, most existing works focus only on constructing a single routing tree. To employ the existing methods of tree construction, the congestion problem must be addressed. The minimum spanning tree (MST) topology leads to the minimum total wirelength, and thus congestion is easier to be controlled than other topologies. However, its topology may result in longer critical paths and thus degrade circuit performance. Though a shortest path tree (SPT) may result in the best performance, its total wirelength (and congestion) may be significantly larger than that constructed by the MST algorithm [10]. In [10], researchers used the idea of incrementally modifying an MST to construct a routing tree for a better trade-off between timing (SPT) and wirelength (MST).

Our construction of a timing-driven routing tree is based on the similar idea used in [10]. We first construct an MST (for smaller wirelength and thus better routability) and then fix the timing violation, if any, by resorting to the SPT topology of the net. Performance optimization usually targets on the minimization of the critical path delay, but to determine a critical path in a circuit is an NP-hard problem due to the false path problem [9]. Therefore, for simplicity, we minimize the critical sink of a net. In the following, we present our framework for timing-driven multilevel routing that is summarized in Figure 5.

The same as the framework for multilevel routing for routability, we first build an MST for each net. However, the MST here is directed since timing analysis is conducted from the tree source to all sinks, opposite to the multilevel routing for routability that uses undirected trees. After the topologies of all nets are obtained, our multilevel framework starts from coarsening the finest tiles at level 0 and processes tiles one by one. Before we route a local net (connection), timing analysis based on the Elmore delay model is performed from the tree source to all sinks. If a target node violates the timing constraint, we modify the tree topology by *recalling modification*. That is, if a target node violates the timing constraint, we delete this local connection and then trace back from the target node to the tree source to find a new parent for the connection that can meet the timing constraint. (Although this process might increase the total wirelength and thus the total wire capacitance, the decrease of the path delay due to lower source-to-sink loading capacitance is even more significant.) Figure 4 shows how to trace back the tree from the target node to the source to find a new node to satisfy the timing constraint. After a new path that meets the timing constraint is found, we start to route the net if it is a local net belonging to the current level. The routing process is the same as that for multilevel routing for routability. After detailed routing is done, the target node may again violate the timing constraint because the detailed route may run through a longer path or incur a larger load from other tree branches. We will fix the timing violation at the later uncoarsening stage. In order to alleviate this problem, we may keep a small timing slack when we estimate the path delay.

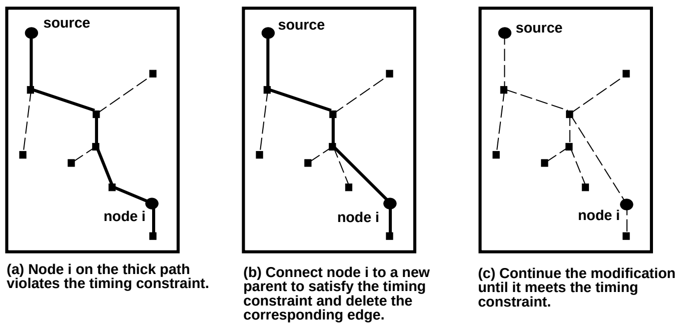
source
node i
(a) Node i on the thick path violates the timing constraint.
source
node i
(b) Connect node i to a new parent to satisfy the timing constraint and delete the corresponding edge.
source
node i
(c) Continue the modification until it meets the timing constraint.

Figure 4: An example of recalling modification.

After coarsening is done, our algorithm performs timing analysis on all nets again to identify those nets that violate the timing constraints.

Uncarsening continues to refine those failed nets, if any, by maze routing. Also, the failed nets from the coarsening stage are refined. Since we iteratively fine tune every local net, a topology of the net meeting the timing constraint and possessing good routability is gradually formed. Like [5], the iterative refinement provides a framework for seamless integration of different algorithms at different levels.

```
Algorithm:  Performance-Driven-Multilevel-Routing(G, N, C)
    Input:  G - partitioned layout;
            N - netlist of multi-terminal nets;
            C - timing constraints.
   Output:  routing solutions for N on G
begin
1   Partition layout and build MSTs for $N$;
2   //coarsening stage
3   For each level at the coarsening stage
4       Choose a local net $n$;
5       if $n$ violates its timing constraint,
            apply recalling modification to fix timing;
6       if $n$ belongs to this level
7           Global_Pattern_Routing();
8           Detailed_Routing();
9   //uncoarsening stage
10  For each level at the uncoarsening stage
11      Timing_Analysis_on_All_Nets();
12      Choose a local net $n$ that violates its timing constraint
            or a failed net during coarsening;
13      if $n$ violates its timing constraint,
            apply recalling modification to fix timing;
14      Global_Maze_Routing();
15      Detailed_Routing();
16  Output_Result();
end
```

Figure 5: Algorithm for performance-driven multilevel routing.

3.2.2 Via Minimization

Vias typically have significantly larger RC delay than metal wires, and thus it is desired to minimize the number of vias used in a routing path to optimize circuit performance. We apply the following algorithm, called *SPVM* (*Simultaneous Pathlength and Via Minimization*), to perform maze routing to find a shortest path with the minimum number of bends/vias. It associates each basic detailed routing region u (could be a grid cell in gridded-based routing or a basic routing region defined by the wire pitch in gridless routing) with two labels: $d(u)$ and $b(u)$, where $d(u)$ is the distance of the shortest path from source s to u, and $b(u)$ is the minimum number of bends/vias along the shortest path from s to u. Initialize $d(u) = \infty, b(u) = \infty, \forall u \neq s, d(s) = 0$ and $b(s) = 0$. Maze routing is a two-stage approach of wave propagation followed by backtracking [16]. In the wave-propagation stage of maze routing, the computation of label d's is the same as the original maze-routing algorithm. Let u be a basic routing region on the wave front and v a neighboring basic routing region of u. The *predecessor routing region* of u is the region from which the wave front was propagated for obtaining the minimum $b(u)$. The propagation direction of u is the direction from the predecessor routing region of u to u. The computation of $b(v)$ is as follows.

The basic idea is to compare the distance label d's first and then compare the bend/via number label b's. The value $b(v)$ of a neighboring routing region v with $d(v) < d(u)$ stays unchanged because the path from s through u to v is not the shortest path between s and v. The backtracking stage is the same as that of the original maze-routing algorithm. Note that it is possible that there may exist several shortest

```
1 if  $(d(v) \geq d(u) + 1)$
2   if  $(b(v) > b(u)$ and  $v$ is along the propagation direction of $u)$
3       $b(v) \leftarrow b(u)$;
4       Record $u$ as the predecessor routing region of $v$;
5   if  $(b(v) > b(u) + 1$ and  $v$ is not along the propagation
        direction of $u)$
6       $b(v) \leftarrow b(u) + 1$;
7       Record $u$ as the predecessor routing region of $v$.
```

Figure 6: The algorithm to compute $b(v)$.

paths with different number of bends/vias. The wave-propagation stage always keeps track of the shortest path with the minimum bend/via number to allow the backtracking stage to find such a path. We have the following theorem.

Theorem 1 *Algorithm SPVM guarantees to find a shortest path with the minimum number of bends/vias, if the path exists.*

4 Experimental Results

We have implemented our multilevel routing system in the C++ language on a 450 MHz SUN Sparc Ultra-60 workstation with 2 GB memory. The routing system is available at the web site http://cc.ee.ntu.edu.tw/∼ywchang/research.html. We compared our results with [5] and [6] based on the 11 benchmark circuits provided by the authors. The design rules for wire/via widths and wire/via separation for detailed routing are the same as those used in [5, 6]. The parameters a and b in the cost function β were both set to 1 while c was initially set to 1 and was gradually increased when the router failed to refine the target net until a termination bound was reached.

Table 2 lists the set of benchmark circuits. In the table, "Ex." gives the names of the circuits, "Size" gives the layout dimensions, "#Layers" denotes the number of routing layers used, and "#Nets" gives the number of two-pin connections after net decomposition. Table 3 gives the comparison of our multilevel routing for routability with the three-level routing [6], the hierarchical routing [5], and the multilevel routing [5]. The three-level routing (A) first uses a performance-driven global router, then a noise-constrained wire spacing and track assignment algorithm, and a detailed router [6]. The hierarchical routing with rip-up and replan (B) is developed in [5] for comparative study. Since the hierarchical approach adopts the top-down process to handle designs, it has a more global view of the problem. But, as mentioned earlier, a hierarchical flow lacks local routing information and needs to refine more local congestion than a multilevel approach does. The multilevel routing (C) gives the main results from [5]. In the table, "Time (s)" represents the running times in second, "#Rtd. Nets" denotes the number of routed nets, "Comp. Rates" gives the routing completion rates, and "avg." (bottom) denotes the average routing completion rates.

As shown in the table, our approach obtains significantly better routing solutions than the multilevel routing [5], the three-level routing [6], and the hierarchical approach [5]. For the 11 benchmark circuits provided by the authors of [5], our approach obtains 100% routing completion for all circuits while the multilevel routing, the three-level routing, and the hierarchical routing can complete routing for only 3, 0, 3 circuits, respectively.

Since all examples are 100% routed by our system using the numbers of layers given in the test data, we show our superior performance by further reducing the numbers of available routing layers in the examples. Table 4 shows that our multilevel router still obtains better routing

Ex.	#Layers	(A) Three-Level Routing			(B) Hierarchical Routing with Rip-up and Re-route			(C) Multilevel Routing of [5]			(D) Our Results		
		Time (s)	#Rtd. Nets	Cmp. Rates	Time (s)	#Rtd. Nets	Cmp. Rates	Time (s)	#Rtd. Nets	Cmp. Rates	Time (s)	#Rtd. Nets	Cmp. Rates
Mcc1	4	933.2	1499	88%	947.9	1600	94.5%	436.7	1683	99.4%	204.7	1694	100%
Mcc2	4	12333.6	5451	72.3%	10101.4	7161	95.6%	7644.8	7474	99.1%	7203.3	7541	100%
Struct	3	406.2	3530	99.4%	324.5	3551	100%	316.8	3551	100%	151.5	3551	100%
Prim1	3	239.1	2018	99.0%	353.0	2037	100%	350.2	2037	100%	165.4	2037	100%
Prim2	3	1331	8109	98.9%	2423.8	8194	100%	2488.4	8196	100%	788.2	8197	100%
S5378	3	430.2	2607	83.4%	57.9	2964	94.9%	54.0	2963	94.8%	10.9	3124	100%
S9234	3	355.2	2467	88.9%	40.7	2564	92.4%	41.0	2561	92.3%	7.7	2774	100%
S13207	3	1099.5	6118	87.5%	161.9	6540	93.5%	188.8	6574	94.0%	38.2	6995	100%
S15850	3	1469.1	7343	88.2%	426.1	7874	94.6%	403.4	7863	94.5%	57.5	8321	100%
s38417	3	3560.9	19090	90.8%	754.6	19596	93.2%	733.6	19636	93.3%	137.6	21035	100%
S38584	3	7086.5	25642	91.0%	1720	26461	93.9%	1721.6	26504	94.1%	316.7	28177	100%
avg.				89.8%			95.7%			96.5%			100%

Table 3: Comparison among (A) the three-level routing [6], (B) the hierarchical routing [5], (C) the multilevel routing [5], and (D) our multilevel routing. Note: (A), (B), (C) were run on a 440 Mhz Sun Ultra-5 with 384 MB memory; (D) was run on a 450Mhz Sun Sparc Ultra-60 with 2GB MB.

Ex.	Size (μm)	#Layers	#Nets	#Pins
Mcc1	39000×45000	4	1694	3101
Mcc2	152400×152400	4	7541	25024
Struct	4903×4904	3	3551	5717
Prim1	7552×4988	3	2037	2941
Prim2	10438×6468	3	8197	11226
S5378	4330×2370	3	3124	4734
S9234	4020×2230	3	2774	4185
S13207	6590×3640	3	6995	10562
S15850	7040×3880	3	8321	12566
S38417	11430×6180	3	21035	32210
S38584	12940×6710	3	28177	42589

Table 2: The benchmark circuits.

Ex.	#Layers = 2			#Layers = 3		
	Time (s)	#Rtd. Nets	Cmp. Rates	Time (s)	#Rtd. Nets	Cmp. Rates
Mcc1	242.1	1686	99.4%	204.7	1694	100%
Mcc2	-	-	-	25189.7	7272	96.4%
Struct	151.5	3551	100%	151.7	3551	100%
Prim1	165.4	2037	100%	166.5	2037	100%
Prim2	788.2	8197	100%	789.6	8197	100%
S5378	24.8	3099	99.1%	10.9	3124	100%
S9234	12.3	2767	99.7%	7.7	2774	100%
S13207	56.6	6979	99.7%	38.2	6995	100%
S15850	164.4	8299	99.7%	57.5	8321	100%
S38417	208.3	21012	99.8%	137.6	21035	100%
S38584	681.2	28122	99.8%	316.7	28177	100%
avg.			(99.7%)*			99.6%

Table 4: Results of our multilevel routing for routability by using two and three layers. (*: exclude the rate for Mcc2.)

completion rates by even using fewer layers. From Table 4, we can see that if we only use two layers, our router often needs more time for performing routing since rip-up and re-route might occur more often as the routing resources become more restricted.

We also performed experiments on timing-driven routing (although no previous timing-driven routers are available to us for comparative studies). In the benchmark circuits, Mcc1, Mcc2, Prim1 and Prim2 do not have the information of net sources. Therefore, we cannot calculate the path delay for those benchmarks, and thus only the results for the six examples listed in Table 5 are reported. To perform experiments on timing-driven routing, we used the same resistance, capacitance, and via parameters as those used in [11]. First, we constructed a shortest path tree for a net by connecting all sinks directly to their net source to obtain the timing constraints. We then assigned the timing bound of each sink as the multiplication of the constant k and the shortest path delay of the net. We tried different values of k's and used three layers for routing. As shown in Table 5, as k approaches 2.5 (2.0), the routing completion rates obtained by our timing-driven multilevel routing system are higher than (comparable to) those obtained in [5] that considered only routability. Further, our timing-driven router can dramatically reduce both the critical path delay (d_{max}) and the average net delay (d_{avg}). Therefore, the performance-driven multilevel router is very promising. Figure 7 shows the 2-layer routing solution for "S9234" obtained from our system with routability consideration alone (completion rates = 99.7%). Figure 8 shows the 3-layer routing solution for "S9234" from our timing-driven multilevel routing with $k = 2$ (completion rates = 94.3%).

5 Conclusion

We have proposed a novel multilevel routing framework considering both routability and performance. Unlike the previous multilevel routing, we have integrated global routing, detailed routing, and resource estimation together at each level of the framework, leading to more accurate routing resource estimation during coarsening and thus facilitating the solution refinement during uncoarsening. The exact routing information at each level makes our framework more flexible in dealing with various routing objectives (such as crosstalk, power, etc). Experimental results have shown that our approach is very promising. Future work lies in the development of a performance-driven multilevel router considering signal integrity.

6 Acknowledgement

We would like to thank the authors of [5], Professor Jason Cong, Mr. Jie Fang, and Ms. Yan Zhang, for providing us with the benchmark circuits. Special thanks go to Ms. Yan Zhang for her prompt explanations of their data and very helpful discussions. Thanks also go to the anonymous reviewers for their constructive comments.

References

[1] C. Albrecht, "Global routing by new approximation algorithms for multicommodity flow," *Trans. on Computer-Aided Design*, vol. 20, no. 5, pp. 622-632, May 2001.

Ex.	$k = \infty$ (Routability Alone)				$k = 2.5$				$k = 2$				$k = 1.5$			
	Time (s)	Cmp. Rates	d_{max} (ps)	d_{avg} (ps)	Time (s)	Cmp. Rates	d_{max} (ps)	d_{avg} (ps)	Time (s)	Cmp. Rates	d_{max} (ps)	d_{avg} (ps)	Time (s)	Cmp. Rates	d_{max} (ps)	d_{avg} (ps)
S5378	11	100%	89057	1850	125	96.4%	13695	914	147	94.6%	13651	798	173	90.2%	9999	685
S9234	8	100%	253151	1894	117	96.3%	11942	748	155	94.3%	11426	659	199	88.8%	9999	568
S13207	38	100%	464387	2384	745	94.7%	24862	873	799	93.1%	20149	749	1115	88.5%	15683	654
S15850	58	100%	2.65e+6	2747	1909	95.2%	34904	965	2074	93.1%	28049	859	2658	88.7%	20360	736
S38417	138	100%	8.51e+6	4528	7676	94.9%	50201	816	8883	93.4%	40500	702	10239	89.5%	37809	605
S38584	317	100%	1.75e+8	13799	12374	94.9%	129289	849	13276	93.7%	129267	739	15209	89.8%	129287	634
avg.		100%				95.4%				93.7%				89.2%		

Table 5: Results of our timing-driven multilevel routing with different constraint ratios k's. *Note: Time (s) includes constraint calculation and timing-driven multilevel routing.*

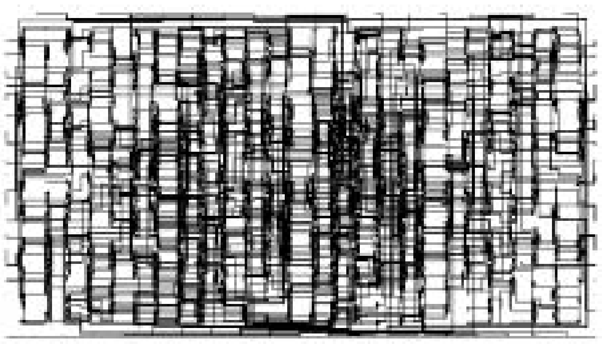

Figure 7: The routing solution for "S9234" obtained from our multilevel routing for routability (2 layers; completion rates = 99.7%).

[2] C. J. Alpert, J.-H. Huang, and A. B. Kahng, "Multilevel circuit partitioning," *IEEE Trans. on Computer-Aided Design*, vol. 17, no. 8, pp. 655–667, August 1998.

[3] Y.-W. Chang, K. Zhu and D. F. Wong, "Timing-driven routing for symmetrical-array-based FPGAs," *ACM Trans. on Design Automation of Electronic Systems*, vol. 5, no. 3, pp. 433-450, July 2000.

[4] T. Chan, J. Cong, T. Kong, and J. Shinnerl, "Multilevel optimization for large-scale circuit placement," *Proc. ICCAD*, pp. 171–176, Nov. 2000.

[5] J. Cong, J. Fang and Y. Zhang, "Multilevel approach to full-chip gridless routing," *Proc. ICCAD*, pp. 396-403, Nov. 2001.

[6] J. Cong, J. Fang and K. Khoo, "DUNE: A multi-layer gridless routing system with wire planning," *Proc. ISPD*, pp. 12-18, April 2000.

[7] J. Cong, A. Kahng, and K. Leung, "Efficient algorithms for the Minimum Shortest Path Steiner Arborescence Problem with Applications to VLSI Physical Design," *Trans. on Computer-Aided DEsign*, vol 17, pp. 24-39, 1998.

[8] J. Cong, S. Lim, and C. Wu, "Performance driven multilevel and multiway partitioning with retiming" *Proc. DAC*, pp. 274–279, June 2000.

[9] J. Cong and P. H. Madden, "Performance driven global routing for standard cell design" *Proc. ISPD*, pp. 73-80, April 1997.

[10] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, "Provably good performancedriven global routing," *Trans. Computer-Aided Design*, vol 11, no 6, pp. 739-752, June 1992.

[11] T. Deguchi, T. Koide and S. Wakabayashi "Timing-driven hierarchical global routing with wire-sizing and buffer-insertion for VLSI with multi-routing-layer," *Proc. ASP-DAC*, pp. 99-104, June 2000.

[12] J. Heisterman and T. lengauer, "The efficient solutions of integer programs for hierarchical global routing," *Trans. on Computer-Aided Design*, vol. 10, no. 6, pp. 748-753, June 1991.

[13] D. Hightower, "A solution to line routing problems on the continuous plane," *Proc. Design Automation Workshop*, pp. 1-24, 1969.

[14] G. Karypis, R. Aggarwal, V. Kumar, and S. shekhar, "Multilevel hypergraph partitioning: Application in VLSI domain," *IEEE Trans. on VLSI Systems*, Vol. 7, pp. 69–79, March 1999.

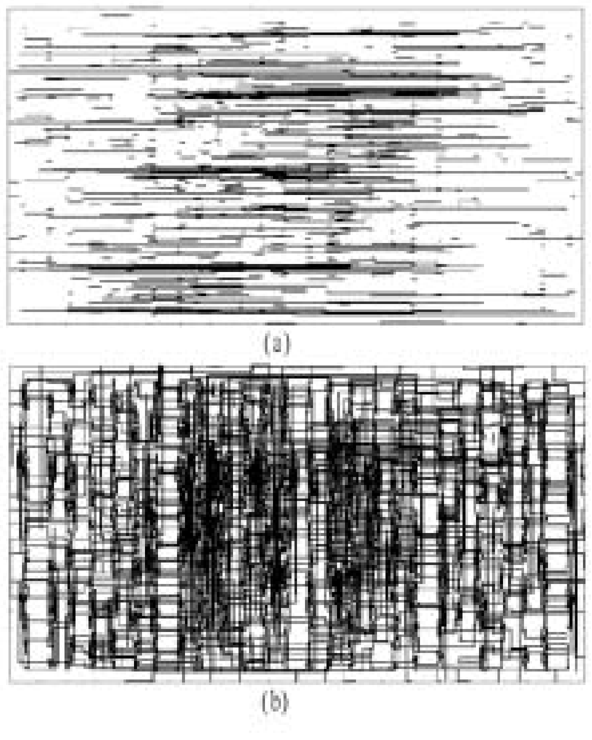

Figure 8: The routing solution for "S9234" obtained from our timing-driven multilevel routing (HVH routing model; α = 2; 3 layers; completion rates = 94.3%). (a) shows the routes on the third horizontal layer and (b) gives the routes on the first and second layers.

[15] R. Kastner, E. Bozorgzadeh and M. Sarrafzadeh, "Predictable routing," *Proc. ICCAD*, pp. 110-114, Nov. 2000.

[16] Lee, "An algorithm for path connection and its application," *IRE Trans. Electronic Computer*, EC-10, 1961.

[17] J. Lillis, C.-K Cheng, T.-T. Y. Lin and C.-Y. Ho, " New Performance Driven routing techniques with explicit area/delay tradeoff and simultaneous wiresizing," *Proc. DAC*, pp. 395-400, June 1996.

[18] S.-C. Lee, J.-M. Hsu, and Y.-W. Chang, "Multilevel large-scale module placement/floorplanning using B*-trees," *Proc. The 12th VLSI Design/CAD Symposium*, Hsinchu, Taiwan, Aug. 2001.

[19] M. Marek-Sadowska, "Router planner for custom chip design," *Proc. ICCAD*, Nov. 1986.

[20] G. Meixner and U. Lauther, "A new global router based on a flow model and linear assignment," *Proc. ICCAD*, pp. 44-47, Nov. 1990.

[21] J. Soukup, "Fast maze router," *Proc. DAC*, pp. 100-102, June 1978.

[22] D. Wang and E. Kuh, "A new timing-driven multilayer MCM/IC routing algorithm," *Proc. Multi-chip Module Conf.*, pp. 89–94, Feb. 1997.