

Short Papers

MR: A New Framework for Multilevel Full-Chip Routing

Yao-Wen Chang and Shih-Ping Lin

Abstract—In this paper, we propose a novel framework for multilevel full-chip routing considering both routability and performance called MR. The two-stage multilevel framework consists of coarsening, followed by uncoarsening. Unlike the previous multilevel routing, MR integrates 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 MR more flexible in dealing with various routing objectives (such as crosstalk, power, etc.). Experimental results show that MR obtains significantly better routing solutions than previous works. For example, for a set of 11 commonly used benchmark circuits, MR 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 2, 0, 2 circuits, respectively. In particular, the number of routing layers used by MR is even smaller. We also have performed experiments on timing-driven routing. The results are also very promising.

Index Terms—Detailed routing, estimation, global routing, layout, physical design, routing, timing optimization.

I. INTRODUCTION

Research in very large scale integrated (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 [17], [24] and line-searching approaches [14], which route net-by-net. Most concurrent algorithms apply network-flow or linear-assignment formulation [1], [23] 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 [22]; Heisterman and Lengauer presented a hierarchical integer linear programming approach for global routing [13]; Wang and Kuh proposed a hierarchical

$(\alpha, \beta)^*$ algorithm for timing-driven multilayer MCM/IC routing [25]; Chang *et al.* applied linear assignment to develop a hierarchical, concurrent global and detailed router for field programmable gate arrays (FPGAs) [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 $2.5 \times 2.5 \text{ cm}^2$ 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* [15], and *HPM* [8], the multilevel placer, *mPL* [4], and the multilevel floorplanner/placer, *MB**-tree [20], all show the promise of the multilevel framework for large-scale circuit partitioning, placement, and floorplanning.

A framework similar to multilevel routing was presented in [12] and [18]. Lin *et al.* in [18] and Hayashi and Tsukiyama in [12] presented hybrid hierarchical *global* routers for multilayer VLSIs [12], in which both bottom-up (coarsening) and top-down (uncoarsening) techniques were used in global routing. Recently, Cong *et al.* proposed a pioneering multilevel approach for large-scale, full-chip, routability-driven (global) 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 multi-commodity 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 global and detailed routing considering both routability and performance called MR. Different from the work presented in [5], MR has the following distinguished features.

- The previous works [5], [12], [18] are mainly for global routing, while our MR integrates global and detailed routing.
- MR integrates 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, MR performs global routing to obtain a good initial solution for all nets inside the

Manuscript received August 16, 2002; revised February 1, 2003. This work was supported in part by the National Science Council of Taiwan, R.O.C., under Grant NSC 91-2215-E-002-038. A preliminary version of this paper was presented at the 2002 IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, November 2002, where it was nominated Best Paper. This paper was recommended by Associate Editor T. Yoshimura.

Y.-W. Chang is with the Department of Electrical Engineering and Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: ywchang@cc.ee.ntu.edu.tw).

S.-P. Lin is with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: is85060@cis.nctu.edu.tw).

Digital Object Identifier 10.1109/TCAD.2004.826547

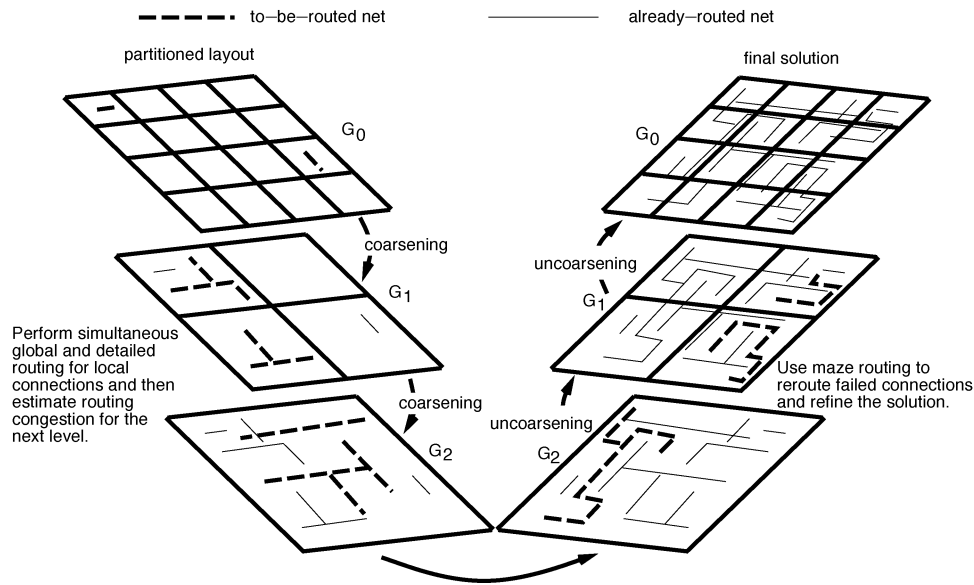


Fig. 1. Multilevel framework flow.

TABLE I
FRAMEWORK COMPARISON BETWEEN OURS AND CONG *et al.* [5]

	Our framework MR	The framework in [5]
Objective	<ul style="list-style-type: none"> • Considers both routability and timing. 	<ul style="list-style-type: none"> • Considers only routability.
Coarsening	<ul style="list-style-type: none"> • Performs global and detailed routing at each level. • Performs congestion estimation after detailed routing. • Uses the Z-shaped routing refinement method. 	<ul style="list-style-type: none"> • Performs only routing resource estimation using a line-sweep algorithm.
After coarsening	<ul style="list-style-type: none"> • No initial routing is needed. 	<ul style="list-style-type: none"> • Initial routing using a multicommodity flow algorithm.
Uncoarsening	<ul style="list-style-type: none"> • Uses a global and a detailed maze routers to refine the solution. 	<ul style="list-style-type: none"> • Uses a global maze router to refine the solution.
Characteristics	<ul style="list-style-type: none"> • 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. 	<ul style="list-style-type: none"> • 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.

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 multilevel processing is finished, the final global routing result is then fed into a detailed router to obtain the final routing solution. It is obvious that MR 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 reroute to refine a routing solution based on the results of resource estimation. If the three tasks are performed separately, the rerouting process conducted at the global routing stage may be in vain since it does not know if the rerouting is useful for the detailed router. Also, the detailed router may fail to find a path because of the low flexibility induced from the sep-

arated 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 rerouting much more effective.
- Unlike the previous works [5], [12], [18] that consider routability alone, MR also applies a recalling modification method to perform timing-driven routing.
- MR 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.

Fig. 1 shows our multilevel framework, and Table I summarizes the differences between MR and that presented in [5].

Experimental results show that MR 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], MR obtains 100% routing

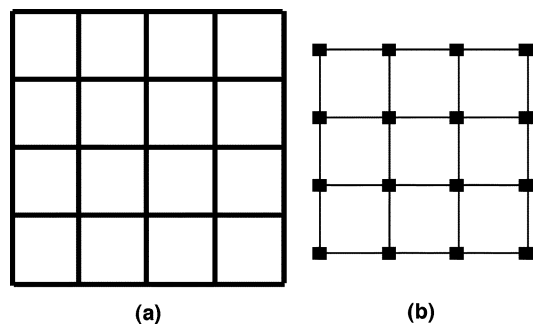


Fig. 2. Routing graph: (a) partitioned layout and (b) routing graph.

completion for all circuits, while the multilevel routing, the three-level routing, and the hierarchical routing can complete routing for only 2, 0, 2 circuits, respectively. In particular, the number of routing layers required for routing completion for MR 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 II presents the routing model and the multilevel routing framework. Section III presents our framework for routability and performance optimization. Experimental results are shown in Section IV. Finally, we give concluding remarks in Section V.

II. PRELIMINARIES

A. Routing Model

Routing in modern ICs is a very complex process 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. Fig. 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 represent 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.

B. Multilevel Routing Model

As illustrated in Fig. 1, G_0 corresponds to the routing graph of the level 0 of the multilevel coarsening stage. At each level, our global router first finds routing paths for the local nets (or local two-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 reroute 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.

III. MULTILEVEL ROUTING FRAMEWORK

Our multilevel routing algorithm, MR, is inspired by the work in [5]. Nevertheless, MR 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 reroute at the lower level because of lacking local routing information. This problem is also with the hierarchical approach.

MR 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 reroute. Thus, this observation implicitly suggests that a shorter net has a higher priority than a longer net as far as routability is concerned. Kastner *et al.* in [16] also suggest this conclusion. Though 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.

A. Multilevel Routing for Routability

Given a netlist, MR first runs the minimum spanning tree (MST) algorithm to construct the topology for each net, and then decomposes each net into two-pin connections, with each connection corresponding to an edge of the minimum spanning tree. MR 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 Fig. 3(a)–(c) for an illustration]. The global routing is based on the approach used in the pattern router [16] 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 $R_e = \{e \in E_i | e \text{ is the edge chosen for routing}\}$. We apply the cost function $\alpha : E_i \rightarrow \mathfrak{R}$ to guide the routing

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

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

$$c_e = \frac{1}{2^{(p_e - d_e)}}$$

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

After the global routing is completed, MR performs detailed routing with the guidance of the global-routing results and finds a real path in the chip. Our detailed router is based on the maze-searching algorithm

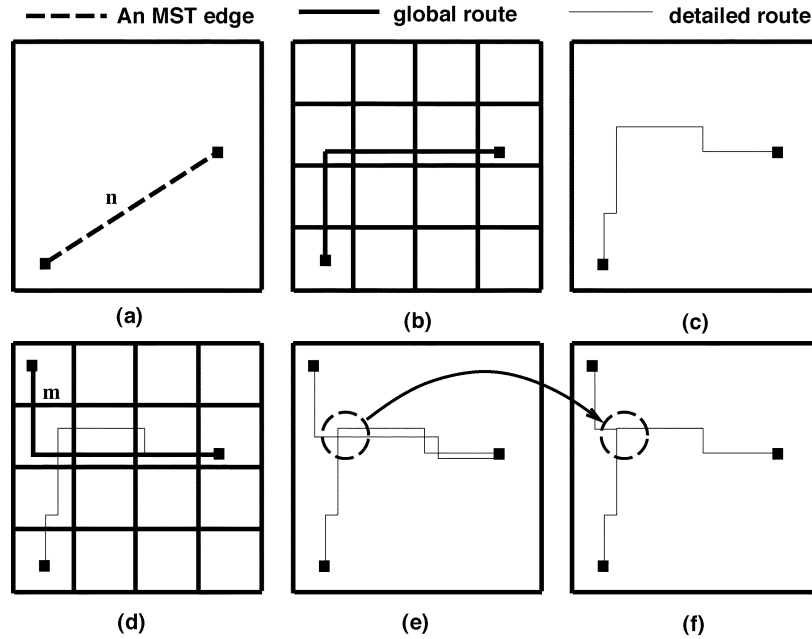


Fig. 3. Global routing, detailed routing, and local refinement. (a) Route the local connection n in a tile of G_i . (b) Global route of n . (c) Detailed route of n on the chip. (d) Route another local connection m that belongs to the same net as n . (e) Detailed route of connection m . (f) Local refinement of net.

and supports the local refinement illustrated in Fig. 3(d)–(f). Pattern routing uses an L- or 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 multi-level 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 reroute. 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- and Z-shaped routing patterns. Fig. 3 shows an example of routing a local net in a tile.

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 $\beta : E_i \rightarrow \mathfrak{R}$:

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

where a – 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 tradeoff 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 reroute processor, in contrast to the simple L- and Z-shaped routing during coarsening. (For rip-up and reroute in MR, we mean the Z-shaped refinement at the coarsening stage, or the maze routing at the uncoarsening stage. They are only applied to global routing for better efficiency and quality tradeoff.) This two-stage approach of global and local refinement of detailed routing gives our overall refinement scheme.

B. Multilevel Routing for Performance

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], [19]. 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 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 tradeoff 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 (see Fig. 4), 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 Fig. 5.

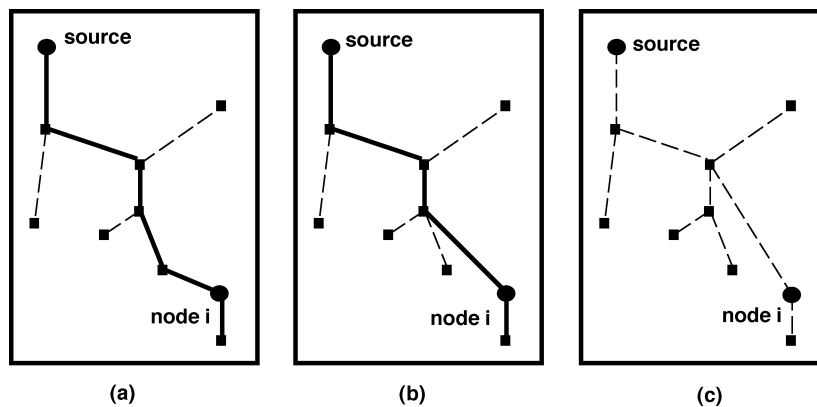


Fig. 4. Example of recalling modification. (a) Node I on the thick path violates the timing constraint. (b) Connect Node I to a new parent to satisfy the timing constraint and delete the corresponding edge. (c) Continue the modification until it meets the timing constraint.

Alg: 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

Fig. 5. Algorithm for performance-driven multilevel routing.

Just as in 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.) Fig. 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

```

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  $v$ .
```

Fig. 6. Algorithm to compute $b(v)$.

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.

After coarsening is done, MR 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.

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 Simultaneous Pathlength and Via Minimization (SPVM), to perform maze routing to find a shortest path with the minimum number of bends/vias (see Fig. 6). 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 [17]. In the wave-propagation stage of maze routing, the computation of label ds is the same as the original maze-routing algorithm. Let u be a basic routing region on the wavefront and v a neighboring basic-routing region of u . The predecessor routing region of u is the region from which the wavefront 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.

TABLE II
BENCHMARK CIRCUITS

Ex.	Size (μm)	#Layers	#Nets	#Pins	#Horizontal/Vertical tracks
Mcc1	39000 \times 45000	4	1694	3101	866/1000
Mcc2	152400 \times 152400	4	7541	25024	3386/3386
Struct	4903 \times 4904	3	3551	5717	2723/2724
Prim1	7552 \times 4988	3	2037	2941	4195/2771
Prim2	10438 \times 6468	3	8197	11226	5798/3593
S5378	4330 \times 2370	3	3124	4734	601/329
S9234	4020 \times 2230	3	2774	4185	558/309
S13207	6590 \times 3640	3	6995	10562	915/505
S15850	7040 \times 3880	3	8321	12566	977/538
S38417	11430 \times 6180	3	21035	32210	1587/858
S38584	12940 \times 6710	3	28177	42589	1797/932

TABLE III
COMPARISON AMONG (A) THE THREE-LEVEL ROUTING [6], (B) THE HIERARCHICAL ROUTING [5], (C) THE MULTILEVEL ROUTING [5], AND (D) OUR MULTILEVEL ROUTING MR. NOTE: (A)–(C) WERE RUN ON A 440-MHZ SUN ULTRA-5 WITH 384 MB OF MEMORY; (D) WAS RUN ON A 450-MHZ SUN SPARC ULTRA-60 WITH 2-GB OF MEMORY

Ex.	# of Layers	(A) Three-Level Routing			(B) Hierarchical Routing with Rip-up and Re-route			(C) Multilevel Routing of [?]			(D) MR		
		Time (s)	#Rtd. Nets	Comp. Rates	Time (s)	#Rtd. Nets	Comp. Rates	Time (s)	#Rtd. Nets	Comp. Rates	Time (s)	#Rtd. Nets	Comp. Rates
Mcc1	4	933	1499	88%	948	1600	94.5%	437	1683	99.4%	205	1694	100%
Mcc2	4	12334	5451	72.3%	101014	7161	95.6%	7645	7474	99.1%	7203	7541	100%
Struct	3	406	3530	99.4%	325	3551	100%	317	3551	100%	152	3551	100%
Prim1	3	239	2018	99.0%	353	2037	100%	350	2037	100%	165	2037	100%
Prim2	3	1331	8109	98.9%	2424	8194	100%	2488	8196	100%	788	8197	100%
S5378	3	430	2607	83.4%	58	2964	94.9%	54	2963	94.8%	11	3124	100%
S9234	3	355	2467	88.9%	41	2564	92.4%	41	2561	92.3%	8	2774	100%
S13207	3	1100	6118	87.5%	162	6540	93.5%	189	6574	94.0%	38	6995	100%
S15850	3	1469	7343	88.2%	426	7874	94.6%	403	7863	94.5%	58	8321	100%
s38417	3	3561	19090	90.8%	755	19596	93.2%	734	19636	93.3%	138	21035	100%
S38584	3	7087	25642	91.0%	1720	26461	93.9%	1722	26504	94.1%	317	28177	100%
avg.				89.8%			95.7%			96.5%			100%

The basic idea is to compare the distance label ds first and then compare the bend/via number label bs . The value $b(v)$ of a neighboring routing region v with $d(v) < d(u)$ remains 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 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. It is clear that the SPVM algorithm guarantees finding a shortest path with the minimum number of bends/vias, if such a path exists.

IV. EXPERIMENTAL RESULTS

We have implemented our multilevel routing system MR in the C++ language on a 450-MHz SUN Sparc Ultra-60 workstation with 2 GB memory. (MR 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] and [6]. The parameters a and b in the cost function β were both set to one, while c was initially set to one, and was gradually increased when the router failed to refine the target net until a termination bound was reached.

Table II lists the set of benchmark circuits. In the table, “Ex.” gives the names of the circuits, “Size” gives the layout dimensions, “# of

Layers” denotes the number of routing layers used, “#Nets” gives the number of two-pin connections after net decomposition, and “#Horizontal/Vertical tracks” gives the number of horizontal/vertical routing tracks per layer. Table III gives the comparison of our multilevel routing MR 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. However, 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 row) denotes the average routing completion rates.

As shown in the table, MR obtains significantly better routing solutions than the multilevel routing [5], the three-level routing [6], and the hierarchical approach [5]. (Note that the previous work [5] also made comparisons with earlier works such as Wang and Kuh [25], which is a maze-based router. As reported in [5], the simple net-by-net maze-based router cannot scale well to handle the circuits used in the experiments.) For the 11 benchmark circuits provided by the authors of [5], MR obtains 100% routing completion for all circuits, while the multilevel routing, the three-level routing, and the hierarchical routing can complete routing for only 2, 0, 2 circuits, respectively.

TABLE IV
RESULTS OF OUR MULTILEVEL ROUTING FOR ROUTABILITY BY USING
TWO AND THREE LAYERS. (*: EXCLUDE THE RATE FOR Mcc2)

Ex.	#Layers = 2			#Layers = 3		
	Time (s)	#Rtd. Nets	Cmp. Rates	Time (s)	#Rtd. Nets	Cmp. Rates
Mcc1	242	1686	99.4%	205	1694	100%
Mcc2	-	-	-	25190	7272	96.4%
Struct	152	3551	100%	152	3551	100%
Prim1	165	2037	100%	167	2037	100%
Prim2	788	8197	100%	790	8197	100%
S5378	25	3099	99.1%	11	3124	100%
S9234	12	2767	99.7%	8	2774	100%
S13207	57	6979	99.7%	38	6995	100%
S15850	164	8299	99.7%	58	8321	100%
S38417	208	21012	99.8%	138	21035	100%
S38584	681	28122	99.8%	317	28177	100%
avg.			(99.7%)*			99.6%

TABLE V
RESULTS OF OUR TIMING-DRIVEN MULTILEVEL ROUTING
WITH DIFFERENT CONSTRAINT RATIOS k s

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

Note: Time (s) includes constraint calculation and timing-driven multilevel routing.

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 IV shows that MR still obtains better routing completion rates by even using fewer layers. From Table IV, we can see that if we only use two layers, MR often needs more time for performing routing, since rip-up and reroute 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 V 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

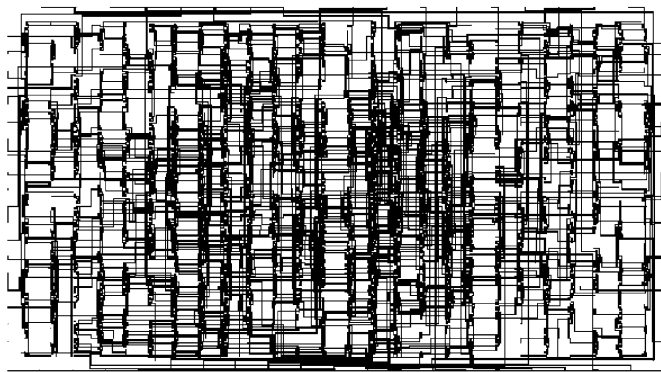
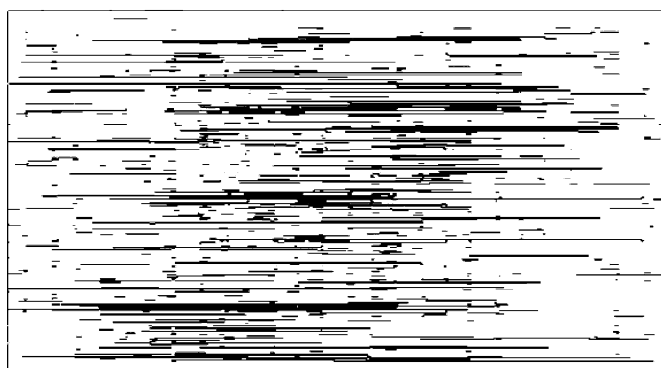
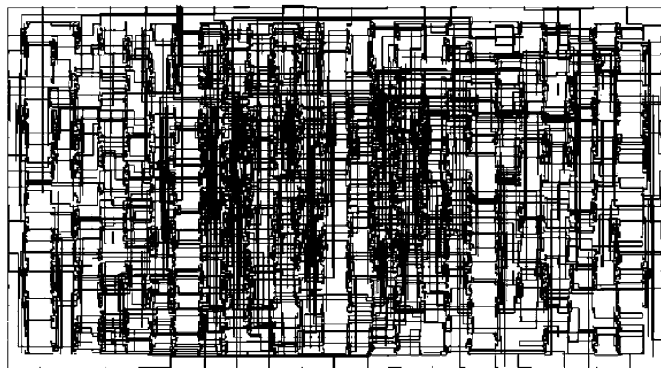


Fig. 7. Routing solution for "S9234" obtained from MR for routability (2 layers; completion rates = 99.7%).



(a)



(b)

Fig. 8. Routing solution for "S9234" obtained from our timing-driven MR (HVH routing model; $\alpha = 2$; 3 layers; completion rates = 94.3%). (a) Routes on the third horizontal layer and (b) routes on the first and second layers.

k s and used three layers for routing. As shown in Table V, as k approaches 2.5 (2.0), the routing completion rates obtained by our timing-driven MR system are higher than (comparable to) those obtained in [5] that considered only routability. Further, our timing-driven MR can dramatically reduce both the critical path delay (d_{max}) and the average net delay (d_{avg}). Therefore, the timing-driven multilevel router MR is very promising. Fig. 7 shows the two-layer routing solution for "S9234" obtained from our system with routability consideration alone (completion rates = 99.7%). Fig. 8 shows the three-layer routing solution for "S9234" from our timing-driven multilevel router with $k = 2$ (completion rates = 94.3%).

The memory requirements ranged from 14 MB for 9234 to 496 MB for s38584, for two-layer routing and were proportional to the number

of layers. For example, for three-layer routing, the circuit s38584 would need about $(496/2) \times 3 = 744$ MB.

V. CONCLUSION

We have proposed a novel multilevel routing framework MR considering both routability and performance. Unlike the previous multilevel routing, MR integrates 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 MR is very promising. Future work lies in the development of a timing-driven multilevel router considering signal integrity.

ACKNOWLEDGMENT

The authors would like to thank the authors of [5], Prof. J. Cong, J. Fang, and Y. Zhang, for providing the benchmark circuits. Special thanks go to Y. Zhang for her prompt explanations of their data and very helpful discussions. They also thank the anonymous reviewers for their very constructive comments.

REFERENCES

- [1] C. Albrecht, "Global routing by new approximation algorithms for multicommodity flow," *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 622–632, May 2001.
- [2] C. J. Alpert, J.-H. Huang, and A. B. Kahng, "Multilevel circuit partitioning," *IEEE Trans. Computer-Aided Design*, vol. 17, pp. 655–667, Aug. 1998.
- [3] Y.-W. Chang, K. Zhu, and D. F. Wong, "Timing-driven routing for symmetrical-array-based FPGAs," *ACM Trans. Design Automation Electron. Syst.*, vol. 5, no. 3, pp. 433–450, 2000.
- [4] T. Chan, J. Cong, T. Kong, and J. Shinnerl, "Multilevel optimization for large-scale circuit placement," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 2000, pp. 171–176.
- [5] J. Cong, J. Fang, and Y. Zhang, "Multilevel approach to full-chip gridless routing," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 2001, pp. 396–403.
- [6] J. Cong, J. Fang, and K. Khoo, "DUNE: A multi-layer gridless routing system with wire planning," in *Proc. ACM Int. Symp. Physical Design*, 2000, pp. 12–18.
- [7] 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. Computer-Aided Design*, vol. 17, pp. 24–39, Jan. 1998.
- [8] J. Cong, S. Lim, and C. Wu, "Performance driven multilevel and multiway partitioning with retiming," in *Proc. ACM/IEEE Design Automation Conf.*, June 2000, pp. 274–279.
- [9] J. Cong and P. H. Madden, "Performance driven global routing for standard cell design," in *Proc. ACM Int. Symp. Physical Design*, Apr. 1997, pp. 73–80.
- [10] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, "Provably good performance driven global routing," *IEEE Trans. Computer-Aided Design*, vol. 11, 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," in *Proc. Asia South Pacific Design Automation Conf.*, June 2000, pp. 99–104.
- [12] M. Hayashi and S. Tsukiyama, "A hybrid hierarchical global router for multi-layer VLSI's," *IEICE Trans. Fundamentals*, vol. E78-A, no. 3, pp. 337–344, 1995.
- [13] J. Heisterman and T. Lengauer, "The efficient solutions of integer programs for hierarchical global routing," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 748–753, June 1991.
- [14] D. Hightower, "A solution to line routing problems on the continuous plane," in *Proc. Design Automation Workshop*, 1969, pp. 1–24.
- [15] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: application in VLSI domain," *IEEE Trans. VLSI Syst.*, vol. 7, pp. 69–79, Mar. 1999.
- [16] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Predictable routing," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 2000, pp. 110–114.
- [17] C.-Y. Lee, "An algorithm for path connection and its application," *IRE Trans. Comput.*, vol. EC-10, pp. 346–365, Sept. 1961.
- [18] Y.-L. Lin, Y.-C. Hsu, and F.-S. Tsai, "Hybrid routing," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 151–157, Feb. 1990.
- [19] 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," in *Proc. Design Automation Conf.*, June 1996, pp. 395–400.
- [20] H.-C. Lee, Y.-W. Chang, J.-M. Hsu, and H. Yang, "Multilevel floorplanning/placement for large-scale modules using B*-trees," in *Proc. ACM/IEEE Design Automation Conf.*, Anaheim, CA, June 2003, pp. 812–817.
- [21] S.-P. Lin and Y.-W. Chang, "Novel framework for multilevel routing considering routability and performance," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 2002, pp. 44–50.
- [22] M. Marek-Sadowska, "Router planner for custom chip design," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1986, pp. 246–249.
- [23] G. Meixner and U. Lauther, "A new global router based on a flow model and linear assignment," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1990, pp. 44–47.
- [24] J. Soukup, "Fast maze router," in *Proc. ACM/IEEE Design Automation Conf.*, June 1978, pp. 100–102.
- [25] D. Wang and E. Kuh, "A new timing-driven multilayer MCM/IC routing algorithm," in *Proc. Multi-Chip Module Conf.*, Feb. 1997, pp. 89–94.

Testing SoC Interconnects for Signal Integrity Using Extended JTAG Architecture

Mohammad H. Tehranipour, Nisar Ahmed, and Mehrdad Nourani

Abstract—As technology shrinks and working frequency reaches the multigigahertz range, designing and testing interconnects are no longer trivial issues. In this paper, we propose an enhanced boundary-scan architecture to test high-speed interconnects for signal integrity. This architecture includes: 1) a modified driving cell that generates patterns according to multiple transitions fault model and 2) an observation cell that monitors signal integrity violations. To fully comply with the conventional Joint Test Action Group Standard, two new instructions are used to control cells and scan activities in the integrity test mode.

Index Terms—Boundary-scan test, integrity loss, interconnect testing, Joint Test Action Group (JTAG) Standard, signal integrity, system-on-chip.

I. INTRODUCTION

A. Motivation

The number of cores in a system-on-chip (SoC) is rapidly growing, which leads to a significant increase in the number of interconnects. With fine miniaturization of the very large scale integrated (VLSI) circuits, existence of long interconnects in SoCs and rapid increase in the working frequency (currently in the gigahertz range), signal integrity

Manuscript received June 23, 2003. This work was supported in part by the National Science Foundation under CAREER Award #CCR-0130513. This paper was recommended by Associate Editor K. Chakrabarty.

The authors are with the Center for Integrated Circuits and Systems, The University of Texas at Dallas, Richardson, TX 75083-0688 USA (e-mail: mht021000@utdallas.edu; nxa018600@utdallas.edu; nourani@utdallas.edu).

Digital Object Identifier 10.1109/TCAD.2004.826540