Crosstalk- and Performance-Driven Multilevel Full-Chip Routing

Tsung-Yi Ho, Yao-Wen Chang, *Member*, *IEEE*, Sao-Jie Chen, *Senior Member*, *IEEE*, and D. T. Lee, *Fellow*, *IEEE*

Abstract

In this paper, we propose a novel framework for fast multilevel routing considering crosstalk and performance optimization. To handle the crosstalk minimization problem, we incorporate an intermediate stage of layer/track assignment into the multilevel routing framework. For performance-driven routing, we propose a novel minimum-radius minimum-cost spanning-tree (MRMCST) heuristic for global routing. Compared with the state-of-the-art multilevel routing with the routability mode, the experimental results show that our router achieved a 6.7X runtime speedup, reduced the respective maximum and average crosstalk (coupling length) by about 30% and 24%, reduced the respective maximum and average delay by about 15% and 5%, and resulted in fewer failed nets.

Keywords

Global Routing, Detailed Routing, Layout, Physical Design, Routing, Timing Optimization, Noise Optimization

I. Introduction

With decreasing feature sizes, higher clock rates, and increasing interconnect densities, crosstalk has become a major concern of comparable importance to area and timing in IC design. Crosstalk profoundly affects the circuit performance in very deep submicron (VDSM) technology; it is introduced by a coupling between two neighboring wires. For example, two adjacent wires form a coupling capacitor. A voltage or a current change on one wire can thus interfere the signal on the other wire. Crosstalk is an unwanted variation which makes the behavior of a manufactured circuit deviate from the expected response. The deleterious influences of crosstalk can be classified into two categories. One is malfunctioning, which makes the logic values of circuit nodes differ from what we desire; the other is timing change, which is caused by switching behavior. Therefore, in addition to routability and timing performance, crosstalk minimization should also be considered in VDSM router design.

Traditionally, the complex routing problem is often solved by 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 [20] and line-searching approaches [16], which route net-by-net. Most concurrent algorithms apply network-flow [1] or linear-assignment formulation [6], [25] to route a set of nets at one time.

The major problem of the flat framework lies in its 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 [25] proposed a hierarchical global router based on linear assignment. Chang, Zhu, and Wong [6] applied linear assignment to develop a hierarchical, concurrent global and detailed router for FPGA's.

The two-level, hierarchical routing framework, however, is still limited in handling the dramatically growing complexity in current and future IC designs. As pointed out in [8], for a 0.07 μm process technology, a 2.5 $\times$ 2.5 cm^2 chip may contain over 360,000 horizontal and vertical routing tracks. To handle such high design complexity, the two-level,

The work of Yao-Wen Chang was partially supported by National Science Council of Taiwan ROC under Grant No.'s NSC91-2215-E-002-038.

Prof. Sao-Jie Chen is with IBM T. J. Watson Research Center during his sabbatical leave; his work was partially supported by the National Science Council of Taiwan ROC under Grant No. NSC 91-2215-E-002-042.

The preliminary version of this paper was presented at the International Conference on Computer Aided Design 2003.

Tsung-Yi Ho is with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan.

Yao-Wen Chang and Sao-Jie Chen are with the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan.

D. T. Lee is with the Institute of Information Science, Academia Sinica, Taipei, Taiwan.

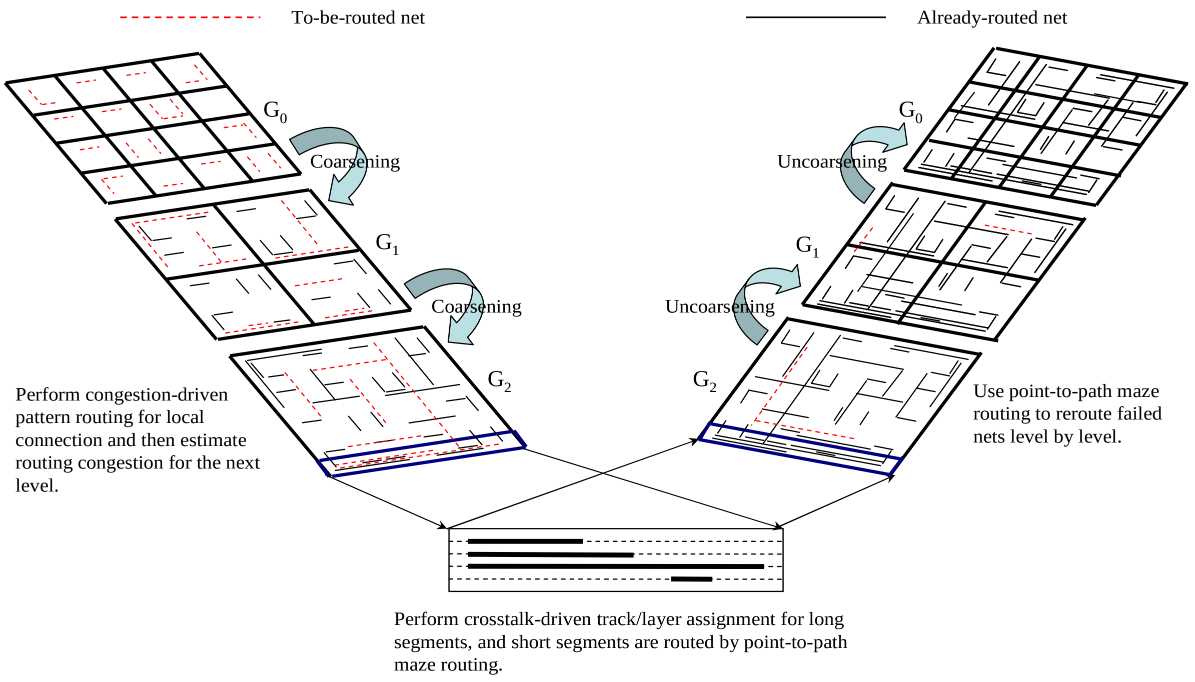

To-be-routed net
Already-routed net
G_0
Coarsening
G_1
Coarsening
G_2
Perform congestion-driven pattern routing for local connection and then estimate routing congestion for the next level.
Uncoarsening
Uncoarsening
Use point-to-path maze routing to reroute failed nets level by level.
Perform crosstalk-driven track/layer assignment for long segments, and short segments are routed by point-to-path maze routing.

Fig. 1. Multilevel framework flow

hierarchical approach becomes insufficient. Therefore, it is desired to employ more levels of routing for very large-scale 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], and *hMETIS* [17], the multilevel placer, *mPL* [4], and the multilevel floorplanner/placer, *MB*-tree* [21], all show the promise of the multilevel framework for large-scale circuit partitioning, placement, and floorplanning.

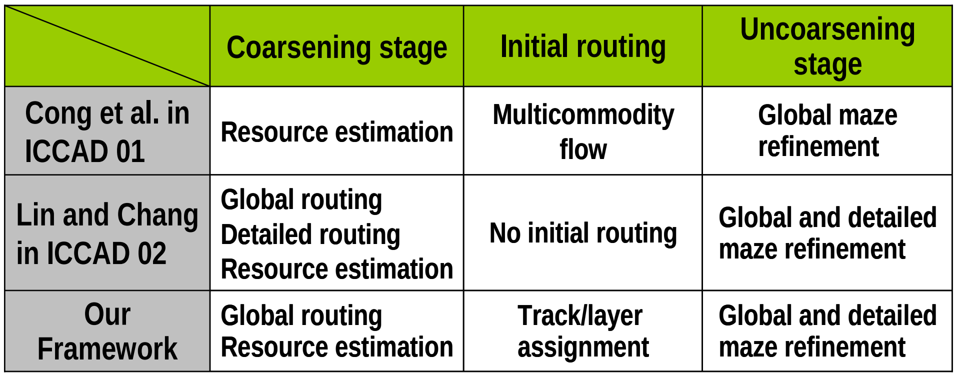
A framework similar to multilevel routing was presented in [15], [23]. Lin, Hsu, and Tsai in [23] and Hayashi and Tsukiyama in [15] presented hybrid hierarchical *global* routers for multi-layer VLSI's [15], in which both bottom-up (coarsening) and top-down (uncoarsening) techniques were used for global routing. Marek-Sadowska [24] proposed a global router based on an outer-most loop approach. The approach is similar to the coarsening stage of multilevel framework. The work first enumerates possible routing patterns for each net in a 2×2 supercell. With the consideration of boundary capacity and routing cost, the mapping of a set of nets to their routing patterns is determined simultaneously. Then, pattern routing is followed by maze routing, and an artificially intelligent rerouter is applied to do rip-up and reroute.

Recently, Cong, Fang, and Zhang proposed a pioneering multilevel global-routing approach for large-scale, full-chip, routability-driven routing [8]. Cong, Xie, and Zhang later proposed an enhanced multilevel routing system, named MARS [9], which incorporates resource reservation, a graph-based Steiner tree heuristic and a history-based multi-iteration scheme to improve the quality of the multilevel routing algorithm in [8]. The final results of both of the multilevel algorithms 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. Lin and Chang also proposed a multilevel approach for full-chip routing, which considers both routability and performance [5], [22]. This framework integrates global routing, detailed routing, and resource estimation together at each level, leading to more accurate routing resource estimation during coarsening and thus facilitating the solution refinement during uncoarsening. Their experimental results show the best routability among the previous works.

Different from the aforementioned works, ours has the following distinguished features:

- A new framework of performing congestion-driven *global* routing at the coarsening stage, followed by an intermediate stage of routing *layer/track assignment* for crosstalk optimization, and then *detailed* routing at the uncoarsening stage. By performing detailed routing after layer/track assignment, we can preserve more flexibility for allocating nets for crosstalk optimization.
- A novel minimum-radius minimum-cost spanning-tree (MRMCST) heuristic is employed to construct routing trees for performance optimization.
- An efficient and effective layer/track assignment scheme is incorporated for crosstalk and run-time optimization.

Figure 1 shows our multilevel framework, and Table 1 summaries the differences of the framework among [8], [22] and ours. Given a netlist, we first run the MRMCST 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 MRMCST. Our multilevel framework starts from coarsening the finest tiles of level 0. At each level, pattern routing is used for routability-driven global routing. After the coarsening stage, we perform a crosstalk-driven layer/track assignment for crosstalk optimization. At the uncoarsening stage, we perform detailed routing. Further, the unroutable nets are performed by point-to-path maze routing [5], [9], [22] and rip-up and re-route to refine the routing solution level by level.



	Coarsening stage	Initial routing	Uncoarsening stage
Cong et al. in ICCAD 01	Resource estimation	Multicommodity flow	Global maze refinement
Lin and Chang in ICCAD 02	Global routing Detailed routing Resource estimation	No initial routing	Global and detailed maze refinement
Our Framework	Global routing Resource estimation	Track/layer assignment	Global and detailed maze refinement

Table 1: Framework comparison between [7], [22] and ours.

Compared with [5], [22] with the routability mode of our router, the experimental results show that our router achieved a 6.7X runtime speedup, reduced the respective maximum and average crosstalk (coupling length) by about 30% and 24%, reduced the respective maximum and average delay by about 15% and 5%. And compared with the timing mode of our router, the experimental results show that our router still achieved a 5.9X runtime speedup, reduced the respective maximum and average crosstalk by about 35% and 23%, reduced the respective maximum and average delay by about 7% and 10% in comparable routability, and resulted in fewer failed nets. The results show the promise of our approach.

The rest of this paper is organized as follows. Section II presents the routing model and the multilevel routing framework. Section III presents our novel framework for run-time and crosstalk optimization. Experimental results are shown in Section IV. Finally, we give concluding remarks in Section V.

II. Preliminaries

A. Routing Model

Our global routing algorithm is based on a graph search technique guided by the congestion information associated with routing regions and topologies. The router assigns higher costs to route nets through congested areas (or those of higher delay and/or crosstalk costs) to balance the net distribution among routing regions.

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 an array of rectangular subregions. These subregions are called *global cells* (*GCs*). A node in the graph represents a *GC* in the chip, and an edge denotes the boundary between two adjacent *GCs*. Each edge is assigned a capacity according to the physical area or the number of tracks of a GC. The graph is used to represent the routing area and is called a *multilevel routing graph*, denoted by G_k, where k is the level number. A global router finds *GC*-to-*GC* paths for all nets on a routing graph to guide the detailed routing. 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 [13]. Wires in each layer run either horizontally or vertically. We refer to the layer as a horizontal (H) or a vertical (V) routing layer.

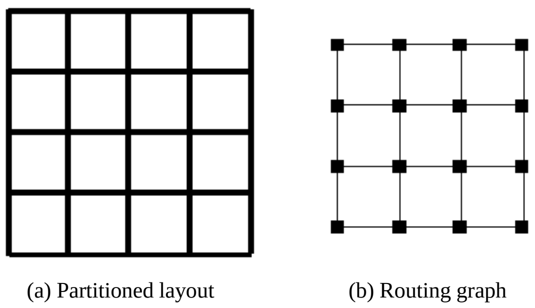


Fig. 2. The routing graph.

B. 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 router first finds routing paths for the *local nets* (or *local 2-pin connections*) (those nets [connections] that entirely sit inside a *GC*). After the global routing is performed, we merge 2×2 *GCs* of G_0 into a larger *GC* 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 *GCs* at a level, say the k-th level, is below a threshold. After the coarsening is finished, a crosstalk-driven layer/track assignment is performed to assign long and straight segments to underlying routing resources. The uncoarsening stage tries to refine the routing solution of the unassigned segments of the level k. During uncoarsening, the unroutable nets are performed by point-to-path maze routing and rip-up and re-route to refine the routing solution. Then we proceed to the next level (level $k-1$) of uncoarsening by expanding each GC_k to four finer GC_{k-1}. The process continues until we reach level 0 when the final routing solution is obtained.

III. Multilevel Routing Framework

Our multilevel routing algorithm is inspired by the work [5], [22]. Nevertheless, our framework is significantly different from that of [5], [22]. Different from the framework of [5], [22] that integrates global routing, detailed routing, and resource estimation together at each level, we perform global routing at the coarsening stage, followed by layer/track assignment at an intermediate stage, and then detailed routing at the uncoarsening stage. At the coarsening stage, a fast congestion-driven pattern routing [18] is used for global routing level by level. After the coarsening stage, we perform layer/track assignment for crosstalk optimization. At this intermediate stage, long and straight segments tend to be assigned to specified layers/tracks, leading to more efficient detailed routing at the uncoarsening stage since often only short segments need to be handled during detailed routing. At the uncoarsening stage, the unroutable nets are routed by point-to-path maze routing and rip-up and re-route to refine the routing solution level by level.

A. Performance-driven Routing Tree Construction

In VDSM IC designs, interconnection delay dominates the performance of a circuit. Therefore, improving the wire delay also improves the overall chip performance. Many techniques have been developed to facilitate high-performance IC designs. For example, the algorithms for constructing performance-driven routing trees have received much attention ([11]). The minimum spanning tree (MST) topology leads to the minimum total wire length, and thus congestion is often easier to be controlled than other topologies. However, its topology may result in longer critical paths and degrade circuit performance. In contrast, a shortest path tree (SPT) may result in the best performance, but its total wire length (and congestion) may be significantly larger than that constructed by the MST algorithm. In [11], researchers used the idea of incrementally modifying an MST to construct a performance-driven routing tree for a smooth trade-off between the tree radius (maximum signal delay) and the tree cost (total interconnection length). On one hand, minimizing wire length minimizes driver's output resistance and the total wire capacitance. On the other hand, minimizing the path length from the source to a sink also minimizes loading capacitance. Thus, both wire length and path length minimization are comparably important for RC delay minimization.

Different from the work presented in [11], our algorithm tries to find a timing-driven routing tree, named a minimum-radius minimum-cost spanning tree (MRMCST), with the minimum radius among all MST's. Since the MRMCST problem is NP-hard [26], we resort to a heuristic to obtain efficient solutions.

Given a vertex v in a graph G, its *eccentricity*, denoted by $ecc(v)$, is the distance from v to the farthest vertex in G, which is also referred to as the *radius* of G with respect to v. The *essential edges* are those contained in every MST, and the *optional edges* are those contained in some MST's but not all MST's. The pseudo-center of a tree, denoted by pc, is a point on an edge or a vertex of *diameter* P of the tree such that the distances from pc to the two extremes of P are the same. By diameter, we mean the longest path between any two vertices in a tree.

Since an MRMCST is an MST with the minimum radius among all possible MST's, this leads us to find the union graph of all MST's (called the MST Union Graph, MSTUG for short) and the intersection graph of all MST's (the MST Intersection Graph, MSTIG for short). We construct an MSTUG and an MSTIG by modifying the edge-coloring process introduced by Tarjan [29], in which edges are colored either blue (essential edges) or red (discarded edges). But neither blue nor red edges can be applied to the optional edges. By modifying the edge-coloring process, we introduce green edges to represent optional edges. The MSTUG contains all the blue and green edges while the MSTIG contains blue edges only.

Initially there are n single components. As edges are colored green or blue, two components are merged together to produce one component. If there exists one and only one component, the algorithm will terminate after coloring all the uncolored edges red. The algorithm is summarized in Figure 3. Based on this modified edge-coloring process, an MSTUG and an MSTIG can be constructed in $O(n \lg n)$ time, where n is the number of vertices. See Figure 4(b) for an example of MSTUG and MSTIG construction.

After constructing an MSTUG and an MSTIG, we may obtain several blue trees and optional edges unless the MST is unique, and then an MRMCST can be constructed by selecting optional edge(s) to connect the blue trees. We introduce a Prim-based heuristic, named *locally optimal connection strategy (LOCS)*, for the MRMCST construction.

The Prim-based method considers only one criterion. If there is more than one minimally cost optional edge incident to the blue tree with source s, we break the tie by choosing the edge $e = (p, q)$, where p is in the blue tree with source s and q is in a neighboring blue tree T, to minimize $f(e, T)$ defined below:

$$f(e,T) = dist(s,p) + cost(e) + dist(q, pc(T)) + ecc(pc(T)), \tag{1}$$

where $dist(s, p)$ is the distance from the source to the node p, and $cost(e)$ is the length of edge e.

The MRMCST algorithm that employs LOCS is summarized in Figure 5. See Figure 4(c) for the MRMCST of Figure 4(b).

Theorem 1 *The MRMCST algorithm runs in $O(n + m_{opt} \lg m_{opt})$ time, where n is the number of vertices and m_{opt} is the number of optional edges.*

```
Algorithm : MSTUG and MSTIG($G$)
    Input : A connected graph $G = (V, E)$;
   Output : Partition $E$ into three sets,
            GREEN (set of optional edges),
            BLUE (set of essential edges), and
            RED (set of discarded edges).
begin
1  Partition the edges of $G$ into equivalence classes
   $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{max}$ s.t. two distinct edges are in the
   same class iff they have the same edge cost.
2  while (there exists more than one component) do
3      For each $e \in \varepsilon_i$
           if both ends of $e$ are in the same current
           component
               then color it RED and delete it from $\varepsilon_i$
4      If $|\varepsilon_i| = 0$ then goto Step 6
           else if $|\varepsilon_i| = 1$ then color it BLUE and
           goto Step 6
           else color them GREEN.
5      For each GREEN edge $e \in \varepsilon_i$
           if $e$ is a bridge in the current component
               then recolor it BLUE.
6      If there is only one component (i.e. connected)
           then color all uncolored edges RED and
           stop.
end
```

Fig. 3. Algorithm for constructing an MSTUG and an MSTIG.

Proof: The merging (connecting) cost is $O(E_i)$ when the blue tree $T_i = (V, E_i)$ is connected to the super blue tree. Hence the total connecting cost will be $O(n)$. Since every optional edge is inserted into the priority queue exactly once and each insertion/deletion for the priority queue needs $O(m_{opt})$ time, the total time complexity for MRMCST construction is $O(n + m_{opt} \lg m_{opt})$, where n is the number of vertices and m_{opt} is the number of optional edges. ■

After MRMCST is constructed, 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 deleting this local connection and then tracing 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.) After all nets meet the timing constraint, we start to route them in the coarsening stage.

B. Crosstalk-Driven Layer/Track Assignment

As fabrication technology shrinks into the VDSM era, as pointed out in [28], on-chip minimum feature sizes continue to decrease, and devices and interconnection wires are placed in closer proximity in order to reduce interconnection delay and routing area. The increasing aspect ratio of wires and the decreasing of interconnect spacing have made the coupling capacitance larger than self capacitance. In fact, the ratio of coupling capacitance is reported to be even as high as $70\% \sim 80\%$ of the total wiring capacitance, even in $0.25\mu m$ technology.

Crosstalk is mostly caused by coupling capacitance between interconnection wires. In general, the crosstalk between two wires is proportional to their coupling capacitance, which is determined by the relative positions of the wires. The coupling capacitance between a pair of parallel wires is proportional to their coupling length, and is inversely proportional to their separating distance. The coupling capacitance between a part of orthogonal wires is negligible in comparison with the coupling capacitance between a pair of parallel wires for current technology. Consequently, it is reasonable to

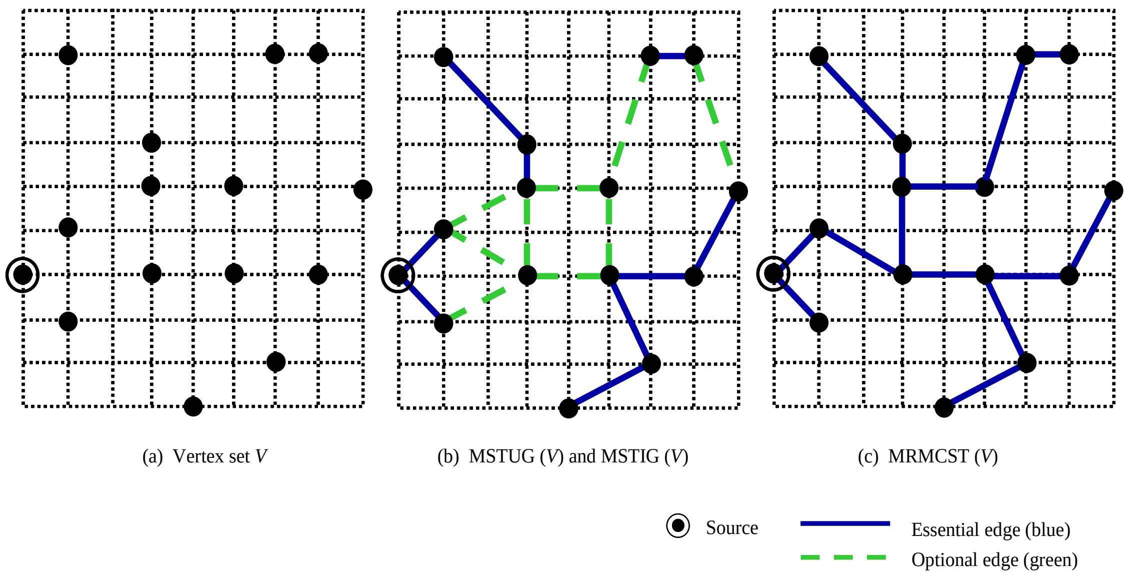


Fig. 4. Example MRMCST construction. (a) The given vertex set. (b) The MSTUG contains all edges while the MSTIG contains all solid edges. (c) The resulting MRMCST.

assume that there is crosstalk only between adjacent parallel wires.

Recently, there has been much research on the coupling problem in both global and detailed routing. Zhou and Wong [31] minimized crosstalk at the global routing stage. Chaudhary et al. [7] proposed wire spacing after detailed routing to reduce crosstalk. This technique can be applied as a post-processing and used for improving an existing layout, but it is not suitable for routing.

However, both global routing and detailed routing are not the best stage to address crosstalk. It might be too early to handle crosstalk during global routing since the relative positions and ordering of nets are not determined at this stage; therefore, the best one can do here is use rough statistical estimators that discourage nets from entering regions where unwanted proximities seem likely. Conversely, it is too late for detailed routing since area routers that embed one net at a time may encounter unsolvable rip-up/re-route problems when trying to embed a late-routing net that must traverse a region already dense with conflicting aggressor or victim nets.

To address these problems, Kay and Rutenbar [19] suggested an integer linear programming (ILP)-based track/layer assignment method to do crosstalk optimization. However, the ILP-based approach is very time-consuming and thus not suitable for large and complex design. Batterywala et al. [3] proposed a fast track assignment heuristic considering routability, but crosstalk was not addressed in the work.

In this paper, inspired by the work [3], we propose a fast layer/track assignment heuristic for crosstalk optimization. After the coarsening stage, we may obtain several long horizontal and vertical segments. To simplify the layer/track assignment problem, we only assign segments which span more than one complete global cell in a row or a column. (We handle short segments during detailed routing.) The layer/track assigner works on a full row or column of the global cell array at a time. Each row (column) is called a *panel*.

We first build the horizontal constraint graph $HCG(V, E)$ for all segments in the panel. Each vertex $v \in V$ corresponds to a segment in the panel. Two vertices v_i and v_j are connected by an edge $e \in E$ iff these segments belong to two different nets and their spans overlap. The edge cost of $e = (v_i, v_j) \in E$ represents the coupling length if v_i and v_j are assigned to adjacent tracks. We define the crosstalk-driven layer assignment problem as follows:

- **The Crosstalk-driven Layer Assignment (CLA) Problem:** Given a set of layers L and a set of segments ℓ, find an assignment of segments to the layers such that the sum of the overlapping lengths among all nets in every layer is minimized.

Here, the cost for CTA comes from the overlapping lengths of nets since nets are not yet assigned to tracks during the

```
Algorithm : MRMCST
   Input  : MSTUG, MSTIG and source $s$
   Output : An MRMCST
begin
1     $T_s$ = a blue tree containing the $s$;
2     $NumOfTrees$ = the number of blue trees;
3     Find the $pc$ for each blue tree which is not $T_s$;
4     For all vertices not in $T_s$, find the distance to
      its corresponding $pc$;
5     For each blue tree $T_i$, find $ecc(pc(T_i))$ and
      optional edges incident to it;
6     For all vertices in $T_s$, find the distance from $s$;
7     Mark all vertices in $T_s$ "visited";
8     For each optional edge $e$ incident to $T_s$, mark it
      "inserted" and call
      InsHeap($e$, FirstCost, SecondCost);
9     while ($NumOfTrees > 1$) do
10        MinE = $(v_1, v_2)$ = PopHeap();
11        while (both $v$ and $w$ are marked "visited") do
12            MinE = $(v_1, v_2)$ = PopHeap();
13        $NumOfTrees--$;
14        $T_s = T_s +$ MinE = $(v_1, v_2)$;
15        mark $v_2$ "visited"; find $dist(v_2, s)$;
16        For each "unvisited" vertex $w$ in the blue
          tree containing $v_2$ (say $T_i$)
17            Update $dist(w, s)$ and mark "visited"
              while traversing $T_i$ from $v_2$;
18        For each "uninserted" optional edge
          $e = (v_3, v_4)$, $v_3 \in T_i$ and $v_4 \in T_j$
19            FirstCost = $cost(v_3, v_4)$;
20            SecondCost = $dist(s, v_3) + cost(v_3, v_4)$
                          $+ dist(v_4, pc(T_j)) +$
                          $ecc(pc(T_j))$);
21            InsHeap($e$, FirstCost, SecondCost);
22            Mark $e$ "inserted";
23    MRMCST = $T_s$;
end
```

Fig. 5. Algorithm for constructing an MRMCST.

layer assignment and all information we have is the spans of nets. The CLA problem can be formulated as the max-cut, k-coloring (MC) problem [27]. However, the MC problem is NP-complete [27]. Thus, we resort to a simple yet efficient heuristic by constructing a maximum spanning tree from the given HCG. Since a tree can be k colored in linear time if we have k layers, we shall first partition the vertices incident on edges with larger costs (coupling lengths) and allocates the corresponding segments to different layers.

Let R be the set of tracks inside a panel. Each track $t \in R$ can be represented by its set of constituent contiguous intervals. Denoting these intervals by x_i, we have $t \equiv \biguplus x_i$. Each x_i is either

- a blocked interval, where no segment from ℓ can be assigned,
- an occupied interval, where a segment from ℓ has been assigned or
- a free interval, where no segment from the set ℓ has yet been assigned.

A segment $seg \in \ell$ is said to be assignable to $t \in R$, $t \equiv \biguplus x_i$, iff $x_i \cap seg \neq \emptyset$ implies that either x_i is a free interval or is an interval occupied by a segment of the same net. We say two nets are *adjacent* if they are assigned to the same layer and placed on two adjacent tracks. Thus, the crosstalk-driven track assignment problem can be defined as:

• **The Crosstalk-driven Track Assignment (CTA) Problem:** Given a set of tracks R and a set of segments ℓ, find an assignment of segments to the tracks such that the sum of the coupling costs (lengths) among adjacent nets of the assignment is minimized.

After layer assignment, most of the edges with larger costs in an HCG are eliminated, and the HCG is decomposed into k subgraphs subHCG$_1$, subHCG$_2$, ..., subHCG$_k$ if we have k layers. Figure 6 shows an example of the track assignment problem for a subHCG, where $\ell = \{a, b, c, d, e, f\}$, $T = \{1, 2, 3, 4\}$, and obstacles on tracks are shaded in grey (e.g., the two obstacles on tracks 3 and 4). We use a bipartite assignment graph to indicate the assignability of segments to tracks. For example, as shown in Figure 6(b), edges between vertex a and vertices 1, 2, and 3 are introduced since segment a can be assigned to track 1, 2, or 3, but not track 4. For easier implementation, we merge the subHCG and the bipartite assignment graph into a combination graph, as shown in Figure 6(c).

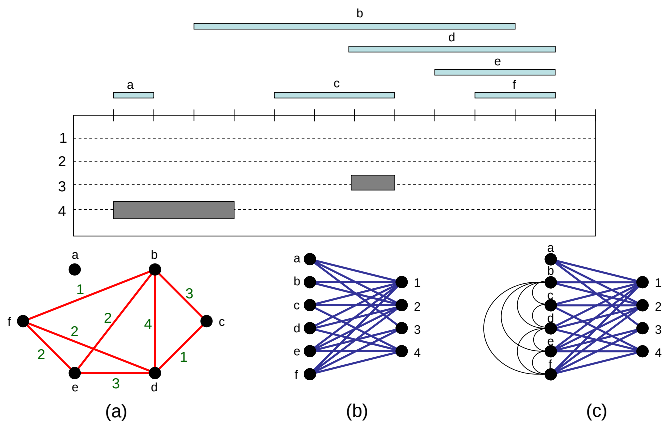


Fig. 6. Constraint graph modeling for track assignment. (a) The subHCG for the given instance. (b) The corresponding bipartite assignment graph. (c) The combination graph.

Since each vertex $v \in V$ corresponds to a segment and each edge $e \in E$ corresponds to the coupling cost in $HCG(V, E)$, the CTA problem can be formulate as the Hamiltonian path problem which has been proven to be NP-complete [12]. We resort to a heuristic for the CTA problem. Our track assignment algorithm starts by finding the maximal sets of conflicting segments. This is equivalent to finding the largest clique V_c in the subgraph subHCG$_i$. Since the HCG graph is an interval graph [14] (a graph induced from interval interactions), finding the largest clique can be done in polynomial time. The algorithm first assigns one maximal subset of conflicting segments at a time by starting from the largest clique. Then we choose the longest segment in the clique as the source s and assign it to the uppermost available track. Then, we choose the min-cost edge (s, i) (and thus the minimal coupling) and assign the segment associated with i to the first available track. If all tracks are occupied, we refer to the net associated with i as a failed net which will be reconsidered at the uncoarsening stage. We repeat the procedure by finding the min-cost edge (i, j) for further processing, where j is an unvisited vertex.

Figure 7(a) shows the final track assignment for the instance of Figure 6. The maximum clique in the subHCG is $\{b, d, e, f\}$, and the longest segment in the clique is b. We thus assign segment b to the uppermost available track, which is track 1. See Figure 7(b) for the updated combination graph after assigning b to track 1. Then, our heuristic makes b the source for constructing the Hamiltonian path for the clique. The min-cost edge $e = (b, f)$ incident on b is chosen, and f is assigned to the first available track. See Figure 7(c) for the updated combination graph after assigning f to track 2. The process is repeated until all vertices in the clique are visited. We then have the track assignment solution shown in Figure 7(a).

After the track assignment, the actual track position of a segment is known. Thus, we can perform point-to-segment maze routing to complete the routing.

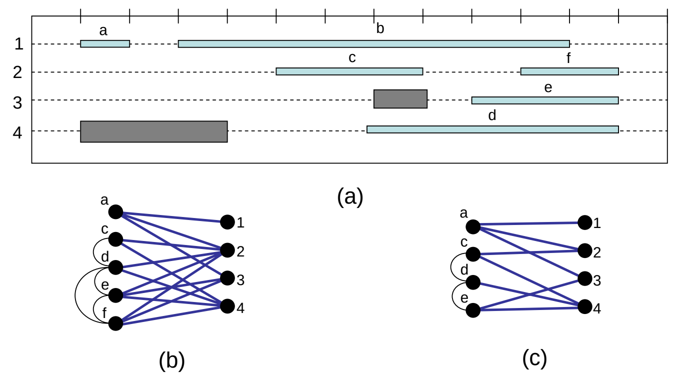


Fig. 7. The process for track assignment. (a) The final track assignment for the instance of Figure 6. (b) The resulting combination graph after assigning b to track 1. (c) The resulting combination graph after assigning f to track 2.

Circuits	Size (μm)	#Layer	#Nets	#Pins
S5378	4330x2370	3	3124	4734
S9234	4020x2230	3	2774	4185
S13207	6590x3640	3	6995	10562
S15850	7040x3880	3	8321	12566
S38417	11430x6180	3	21035	32210
S38584	12940x6710	3	28177	42589

Table 2: The benchmark circuits.

IV. Experimental Results

We have implemented our crosstalk-driven multilevel system in the C++ language on a 1 GHz SUN Blade 2000 workstation with 1GB memory. We compared our results with [5], [22] based on the six benchmark circuits provided by the authors. See Table 1 for the benchmark circuits. (Note that the benchmark circuits used in [5], [8], [22] also contain Mcc1, Mcc2, Struct, Prim1 and Prim2. However, as pointed out in [5], [22], those circuits do not have the information of net sources, and thus we cannot calculate the delay for nets for those benchmarks. Therefore, we shall focus our comparative studies on the six benchmark circuits listed in Table 2.) The design rules for wire/via widths and wire/via separation for detailed routing are the same as those used in [5], [8], [22].

Table 2 describes the set of benchmark circuits. In the table, "Size" gives the layout dimensions, "#Layers" denotes the number of routing layers used, "#Nets" represents the number of two-pin connections after net decomposition. Since the results reported in [5], [22] are better than those in [10] and [8], we shall compare our multilevel router with that in [5], [22].

Circuits	Results of [22]						Our Results					
	D_{max}	D_{avg}	C_{max}	C_{avg}	Time (s)	Cmp. Rates	D_{max}	D_{avg}	C_{max}	C_{avg}	Time (s)	Cmp. Rates
S5378	37308	1403	507	25.4	35	99.7%	27577	1258	342	20.2	10.6	99.8%
S9234	25512	1072	579	21.7	26.2	99.7%	23591	1009	426	17.8	8.1	99.9%
S13207	55337	1262	1526	29.2	106.7	99.8%	52034	1243	1211	22.7	22.6	99.8%
S15850	76297	1302	2913	28.3	538.8	99.3%	68317	1253	2274	22.9	62.6	99.7%
S38417	121419	1170	5704	25.6	899.9	99.5%	105575	1146	4732	20.9	71.3	99.8%
S38584	150936	1208	23196	26.8	1953.7	99.6%	131877	1151	18810	22.6	255.6	99.8%
Comp.	1.15	1.05	1.30	1.24	6.7	1	1	1	1	1	1	+0.2%

Table 3: Results on delay, crosstalk, run-time, and routing completion rate with comparable routability.

Circuits	Results of [22]						Our Results					
	D_{max}	D_{avg}	C_{max}	C_{avg}	Time (s)	Cmp. Rates	D_{max}	D_{avg}	C_{max}	C_{avg}	Time (s)	Cmp. Rates
S5378	13651	798	507	24.7	38.6	94.6 %	12854	751	331	20.3	11.2	95.2 %
S9234	11426	659	579	21.0	27.8	94.3 %	10019	599	426	17.6	8.6	94.2 %
S13207	20149	749	1413	27.7	113.5	93.1 %	18769	693	1109	20.1	24.1	94.4 %
S15850	28049	859	3211	25.9	558.8	93.1 %	25221	743	2510	21.9	73.7	93.6 %
S38417	40500	702	5722	24.6	944.5	93.4 %	38957	670	4385	20.4	90.9	93.7 %
S38584	129267	739	24268	26.7	2187.1	93.7 %	129267	655	17613	22.1	357.0	94.0 %
Comp.	1.07	1.10	1.35	1.23	5.9	1	1	1	1	1	1	+0.4%

Table 4: Results on delay, crosstalk, run-time, and routing completion rate with comparable routability in timing-mode comparison.

Circuits	Results with only CLA		Results with only CTA		Our Results	
	C_{max}	C_{avg}	C_{max}	C_{avg}	C_{max}	C_{avg}
S5378	355	20.2	416	22.4	342	20.2
S9234	480	18.1	501	20.1	426	17.8
S13207	1312	23.6	1373	24.9	1211	22.7
S15850	2398	24.8	2368	24.8	2274	22.9
S38417	4780	23.8	4732	24.0	4732	20.9
S38584	18136	22.9	19618	23.3	18810	22.6
Comp.	+4.6%	+4.4%	+10.2%	+10.0%	–	–

Table 5: Results of crosstalk comparisons.

To perform experiments on timing-driven routing, we used the same resistance and capacitance parameters as those used in [5], [22]. 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. A via is modeled as the Π-model circuit, with its resistance and capacitance being twice of those of a wire segment, and the Elmore delay model is used for our delay computation. All the parameters were the same as those used in [5], [22], and both routers were run on the same machine. Experimental results on run-time, routing completion rate, delay, and crosstalk with comparable routability (for routability optimization) are listed in Table 3. (Note that we set the timing constraint ratio k used in [5], [22] to 5.5 to obtain comparable routability with ours for fair comparisons.) And the results of timing-driven routing with comparable routability are listed in Table 4. (For this experiment, k is set to 2 for [5], [22].) In the table, "D_{max}" represents the critical path delay, "D_{avg}" represents the average net delay, "C_{max}" represents the maximum coupling length of a net, and "C_{avg}" represents the average coupling length. Compared with [5], [22] with the routability mode, the experimental results show that our router achieved a 6.7X runtime speedup, reduced the respective maximum and average crosstalk (coupling length) by about 30% and 24%, reduced the respective maximum and average delay by about 15% and 5%. And compared with the timing-driven mode ($k = 2$ for [5], [22]), the experimental results show that our router still achieved a 5.9X runtime speedup, reduced the respective maximum and average crosstalk by about 35% and 23%, reduced the respective maximum and average delay by about 7% and 10% in comparable routability, and resulted in fewer failed nets.

The results reveal the effectiveness of the intermediate stage of layer and track assignments and our MRMCST for performance-driven routing tree construction. Since many segments are routed in the layer/track assignment stage (which is very efficient), the search space during the uncoarsening stage is significantly reduced. Consequently, the running time and solution quality can be improved simultaneously. Also, compared with [5], [22] that are based on the classical performance-driven routing tree construction, the experimental results on timing have shown that our MRMCST leads to significantly better maximum and average delays.

It should be noted that the coupling capacitance is not included in delay computation for fair comparison with [5], [22]. If coupling capacitance is considered, our router shall able to obtain even better timing reduction due to the significant crosstalk reduction.

To demonstrate the effectiveness of the heuristics used in crosstalk-driven layer assignment (CLA) and track assignment (CTA), we also conducted the following two experiments: First, we performed CLA only for crosstalk minimization, and then the track assignment greedily without considering the cost of the coupling length; second, we simply assigned longer segments to lower layers and then performed CTA for crosstalk minimization. The results are compared to that reported above by minimizing crosstalk using both CLA and CTA. As shown in Table 5, performing CLA and CTA together can reduce the respective coupling costs by 4.6% (4.4%) and 10.2% (10.0%), compared with the results obtained by performing CLA and CTA alone.

V. Conclusion

In this paper, we have proposed a novel framework for fast multilevel routing considering crosstalk and timing optimization. The experimental results have shown that our algorithm is very efficient and effective. Our future work lies in multilevel routing considering other nanometer electrical effects such as antenna avoidance.

References

[1] C. Albrecht, "Global routing by new approximation algorithms for multicommodity flow", *IEEE Trans. on CAD*, vol. 20, no. 5, pp. 622-632, May 2001.
[2] C. J. Alpert, J. H. Huang, and A. B. Kahng, "Multilevel circuit partitioning", *IEEE Trans. on CAD*, vol. 17, no. 8, pp. 655–667, Aug. 1998.
[3] S. H. Batterywala, N. Shenoy, W. Nicholls, and H. Zhou, "Track assignment: A desirable intermediate step between global routing and detailed routing", *Proc. ICCAD*, pp. 59-66, Nov. 2002.
[4] T. F. Chan, J. Cong, T. Kong, and J. R. Shinnerl, "Multilevel optimization for large-scale circuit placement", *Proc. ICCAD*, pp. 171–176, Nov. 2000.
[5] Y. W. Chang and S. P. Lin, "MR: A new framework for multilevel full-chip routing", *IEEE Trans. CAD*, vol. 23, no. 5, May 2004.
[6] Y. W. Chang, K. Zhu, and D. F. Wong, "Timing-driven routing for symmetrical-array-based FPGAs", *Trans. on Design Automation of Electronic Systems*, vol. 5, no. 3, pp. 433-450, July 2000.
[7] K. Chaudhary, A Onozawa, and E. S. Kuh, "A spacing algorithm for performance and crosstalk reduction", *Proc. ICCAD*, pp. 697–702, Nov. 1993.
[8] J. Cong, J. Fang and Y. Zhang, "Multilevel approach to full-chip gridless routing", *Proc. ICCAD*, pp. 396-403, Nov. 2001.
[9] J. Cong, M. Xie and Y. Zhang, "An enhanced multilevel routing system", *Proc. ICCAD*, pp. 51-58, Nov. 2002.
[10] J. Cong, J. Fang and K. Khoo, "DUNE: A multi-layer gridless routing system with wire planning", *Proc. ISPD*, pp. 12-18, Apr. 2000.
[11] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, "Provably good performance-driven global routing", *IEEE Trans. on CAD*, vol 11, no 6, pp. 739-752, Jun. 1992.
[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, 2001.
[13] 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, Jan. 2000.
[14] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, 1980.
[15] 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.
[16] D. Hightower, "A solution to line routing problems on the continuous plane", *Proc. DAW*, pp. 1-24, 1969.
[17] 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, Mar. 1999.
[18] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern routing: use and theory for increasing predictability and avoiding coupling ", *IEEE Trans. on CAD*, pp. 777-790, Nov. 2002.
[19] R. Kay and R. A. Rutenbar, "Wire packing: A strong formulation of crosstalk-aware chip-level track/layer assignment with an efficient integer programming solution", *Proc. ISPD*, pp. 61-68, Apr. 2000.
[20] C. Y. Lee, "An algorithm for path connection and its application", *IRE Trans. Electronic Computer*, EC-10, 1961.
[21] S. C. Lee, Y. W. Chang, J. M. Hsu, and H. Yang, "Multilevel large-scale module floorplanning/placement using B*-trees", *Proc. DAC*, pp. 812-817, Jun. 2003.
[22] S. P. Lin and Y. W. Chang, "A novel framework for multilevel routing considering routability and performance", *Proc. ICCAD*, pp. 44-50, Nov. 2002.
[23] Y. L. Lin, Y. C. Hsu, and F. S. Tsai, "Hybrid routing", *IEEE Trans. on CAD*, Vol. 9, No. 2, pp. 151–157, Feb. 1990.
[24] M. Marek-Sadowska, "Global Router for gate array", *Proc. ICCD*, pp. 332-337, Oct. 1984.
[25] M. Marek-Sadowska, "Router planner for custom chip design", *Proc. ICCAD*, Nov. 1986.
[26] D. Y. Seo and D. T. Lee, *On the complexity of bicriteria spanning tree problems for a set of points in the plane*, PhD disstertation in Northwestern university, 1999.
[27] M. Sriram, S. Kang, J. D. Cho, and S. Raje, and M. Sarrafzadeh, "Crosstalk-minimum layer assignment", *Proc. CICC*, pp. 29.7.1 - 29.7.4, May. 1993.
[28] D. Sylvester, C. Hu, O. Nakagawa, and S.-Y. Oh, "Interconnect scaling: Signal integrity and performance in future high-speed CMOS designs", *Proc. VLSI Symposium on Technology*, 1998.
[29] R. E. Tarjan, *Data structures and network algorithms*, CMBS 44, SIAM, 1983.
[30] T. Y. Ho, Y. W. Chang, S. J. Chen, and D. T. Lee, "A Fast Crosstalk- and Performance-Driven Multilevel Routing System", *Proc. ICCAD*, pp. 382-387, Nov. 2003.
[31] H. Zhou and D. F. Wong, "Global routing with crosstalk constraints", *Proc. DAC*, pp. 374-377, Jun. 1998.

- **Reply to Reviewer 1**

– **R1Q1:** A new framework ... : The main difference of this work (compared to [21]) is that a crosstalk-driven track/layer assignment algorithm is applied on the coarsest level of the multilevel framework. There are also some additional minor differences such as using only global routing during uncoarsening, instead of global+detailed routing as in [21], etc.

– **R1A1:** Thank you very much for the comments. To address nanometer electrical effect problems (e.g. crosstalk), as mentioned in the paper, neither global routing nor detailed routing is the best stage to handle them. It might be too early to handle them during global routing since the relative positions and ordering of nets are not determined at this stage. Conversely, it is too late for detailed routing since area routers that embed one net at a time may encounter unsolvable rip-up/re-route problems when trying to embed a late-routing net that must traverse a region already dense. Our multilevel routing framework uses track/layer assignment as the intermediate step between the coarsening and the uncoarsening stages. In the coarsening stage, we do only global routing. Therefore, there is still much room for avoiding the electrical effects in the intermediate stage because nets are not fixed. Further, we can precisely handle the electrical effects in the uncoarsening stage by detailed routing the nets. Therefore, our new framework is quite suitable to address nanometer electrical effects, and this framework is general and thus is applicable to problems of this kind.

– **R1Q2:** A novel min-radius min-cost spanning-tree (MRMCST) heuristic: It is not very convincing that this heuristic (i.e. minimizing the radius instead of focusing on critical segments) is better than the one used in [21] for performance optimization. In the experimental results, it is claimed that maximum delay is 15% better than the results of [21]. Here, authors have set the parameter k of [21] to 5.5, saying that this gives comparable routability results. (Note that k defines the trade-off between routability and performance in the algorithm of [21]). However, if k were set smaller (e.g. 2.5) in [21], then the maximum delay values in this paper would be about twice as much as the ones in [21], with completion rates only about 5% better (This can be seen from the experimental results of [21]). So, in my opinion, it is not fair to say that this algorithm performs better than [21] in terms of performance optimization.

– **R1A2:** Thank you very much for the comment. Originally, our multilevel routing solution was fixed because we only used MRMCST for performance-driven routing tree construction without setting timing constraints. Thus, we compared with the results of [22] ([21] in the original version) by setting the parameter k to 5.5 for comparable routability. In this revision, we have modified the MRMCST to meet required arrival time by the recalling modification method for better delay reduction. The experimental results by setting $k = 2$ in [22] are shown in Table 4, and the conclusion is about the same as that of the original version.

– **R1Q3:** A point-to-path maze searching algorithm: The details of this algorithm is not given in the paper, but I believe that a similar algorithm is also used in [8].

– **R1A3:** We have added the references to [5], [9], [22] for the point-to-path maze-searching algorithm. In fact, it is kind of straightforward in our implementation. Basically, we keep the net ID number. Expanding from a point, we connect to a path directly if the current search path belongs to the same net as the encountered path.

– **R1Q4:** Layer/track assignment scheme: It is not clearly stated which parts of these algorithms are the contributions of this paper. For example, some parts of the CTA algorithm seem to be similar to the algorithms used in [3] (Although [3] focuses on routability, it is mentioned in [3] that a cost metric for crosstalk avoidance can easily be constructed during track assignment).

– **R1A4:** Thank you very much for the comment. We have made the reference to [3] to be more specific in the new version. The authors in [3] just mention that their approach can be applied to crosstalk avoidance during the track assignment phase without proposing any algorithm to solve it. In this paper, we formulate the CTA problem as a Hamiltonian-path problem and resort to a heuristic to solve it. The experiment shows our algorithm is very effective.

– **R1Q5:** In my opinion, the main contribution of this paper is to apply a crosstalk-driven track/layer assignment algorithm on the coarsest level of the multilevel framework. Other than that, this algorithm is not significantly different (or better) than [21]. Also in [21], it is mentioned that "the framework is flexible and thus different routing objectives (such as crosstalk, power, etc.) can be incorporated since exact track and wiring information at each level is known".

Because of this, I worry that applying a crosstalk-driven algorithm on the coarsest level might not be a substantial contribution by itself.

– **R1A5:** We think the most important contribution of this paper lies in the intermediate stage of the layer/track assignment between the multilevel coarsening and multilevel uncoarsening. As pointed in the paper, this change to the classical multilevel routing framework substantially improve the flexibility and provide a better trade-off in handling many nanometer electrical effects such as crosstalk.

We have added Table 1 in the new version in Page 3 to illustrate the differences in the frameworks among [7], [22] and ours. For our framework, all the nets are globally-routed at the coarsening stage; therefore, we can fully utilize the information of a global router to apply a crosstalk-driven algorithm on the coarsest level. Besides, there is still much room for avoiding the electrical effects in the intermediate stage because nets are not fixed. (If we perform detailed routing at the coarsening stage, then most nets will be fixed with only very little room for crosstalk optimization.) So, we can precisely handle the electrical effects in the uncoarsening stage by routing the nets using a detailed router. Therefore, our new framework is very suitable to address the nanometer electrical effects that need to consider the interactions of nets.

– **R1Q6:** Another point is that in the experimental results section, it is stated that ”Since the results reported in [21] are better than those in [9] and [7], we shall compare our multilevel router with that in [21]”. However, [21] was not compared to [9] and [7] in terms of performance and crosstalk, which are claimed to be the main improvements in this paper. Furthermore, [21] is not crosstalk driven, so I don’t think it is fair to compare the crosstalk results with [21] (Actually, I am not sure whether the detailed routing algorithm used in [21] considers crosstalk, or not. If it is considered, my last comment does not apply.)

– **R1A6:** [22] ([21] in the original version) considers only routability and performance. In this paper, we proposed the first multilevel router that considers both performance and crosstalk. Although the multilevel router of [22] doesn’t consider crosstalk, we compared all metrics (max. delay, average delay, crosstalk) based on comparable routability to [22] for fair comparison. (We believe that this is the best we can do.) The results show that our method achieved a 6.7X runtime speedup, reduced the respective maximum and average crosstalk (coupling length) by about 30% and 24%, reduced the respective maximum and average delay by about 15% and 5%, and resulted in fewer failed nets.

- **Reply to Reviewer 2**

– **R2Q1:** This paper proposes a multilevel routing framework contains three phase: global routing during coarsing, crosstalk-driven layer/track assignment, detailed routing during uncoarsing. Overall, I believe that this is a good flow.

– **R2A1:** Thank you very much for the kind comment.

– **R2Q2:** MRMCST considers minimum radius minimum spanning tree. Question is MRMCST high-performance? There are better tree topologies available for high-performance, why do the authors choose MRMCST? The authors should provide justification. In my opinion, this is the weakest link of the paper.

– **R2A2:** Thank you very much for the comment. Our MRMCST tries to find the tree with the minimal critical path while preserving the minimum total wirelength. This formulation is good for routability and performance trade-off because the wirelength not only consumes routing resource (routability), but also incurs capacitance load. To explicitly consider the timing constraint, we can use MRMCST as an initial solution and apply a local improvement technique to fix timing violations. In this revision, for example, we have modified the MRMCST to meet the required arrival time by the recalling-modification method mentioned in the paper to fix timing violations.

We added a new experiment on performance-driven routing in the revision. Compared with [22] that is based on the classical performance-driven routing tree construction, the experimental results on performance-driven routing (Table 4) have shown that ours outperforms the classical approach by a large margin.

– **R2Q3:** The track assignment is quite restrictive. In Fig. 6, for example, can we divide seg b to two parts, each of which occupies a part of a track?

– **R2A3:** Thank you very much. Although we can divide segments into smaller sub-segments to increase routability, the complexity of the track assignment and the crosstalk handling will get worse. Therefore, we use the whole segments for track assignment to get better results on runtime and crosstalk.

– **R2Q4:** The authors should design experiments to demonstrate the effectiveness of heuristics used in CLA and CTA.

– **R2A4:** Thank you very much for the comments. In the revision, we compared the results with performing both CLA and CTA, CLA alone, and CTA alone. The results are reported in Table 5; the results show that CLA and CTA combined can reduce crosstalk by about 4.5% and 10.1% over those with CLA and CTA alone, respectively.

– **R2Q5:** The authors perhaps should define the radius of a tree, given that they extensively discuss about the diameter of a tree. In fact, I find the discussion on diameter more confusing than helping me with the understanding of the manuscript.

– **R2A5:** Thank you very much. We have added the definition of the radius and modified the content of the paper.

– **R2Q6:** In Fig. 3, the last two lines of the algorithm, "If there is only one component, then...". Is it within the For loop?

– **R2A6:** Thank you very much. We have modified the figure.

– **R2Q7:** In Eqn. (1), is ecc(pc(T)) fixed? If so, it does not affect the optimization.

– **R2A7:** No, the ecc(pc(T)) is not fixed. Because it may have several neighboring blue trees incident to the blue tree with source s, we need to keep the information of every ecc(pc(T)) for LOCS.

– **R2Q8:** Authors maybe need to explain details on fig. 5. (1) Line 15: find dist(v2,s). Is it necessary?

– **R2A8:** Yes, it is necessary. By keeping dist(v2,s), we can perform LOCS in constant time.

– **R2Q9:** (2) Line 17: dist(w,c). What is c?

– **R2A9:** Thank you very much. We have corrected it to s which is the source.

– **R2Q10:** In page 9, the authors should provide the definition of x_i, t, seg, and $x_i \cap seg = \emptyset$, for example, through a figure.

– **R2A10:** Thank you very much for the suggestion. We have provided the definition of the x_i, t, seg, and $x_i \cap seg = \emptyset$ on page 8 in the paper.

– **R2Q11:** In page 9, the authors should illustrate how CTA can be formulated as Hamiltonian path problem.

– **R2A11:** Thank you very much for the suggestion. We have added it in the revision.

– **R2Q12:** In page 9, "Since the graph is an interval graph, finding the largest clique Vc in the subgraph can be done in polynominal time." The authors might be assuming too much about the background of the reader. Perhaps, a brief description of the interval graph would be useful. (same problem as 6).

– **R2A12:** Thank you very much for the suggestion. We have added it in the revision and referred to [14].

• **Reply to Reviewer 3**

– **R3Q1:** This work addresses an important practical problem, that is routing considering timing and crosstalk. A multi-level approach is employed, this is a promising technique for such high problem complexity. Authors claim the major contribution are the MRMCST and track/layer assignment, but the former is provided without explaining why it is necessary, the latter is an easy greedy heuristic that is trivial.

– **R3A1:** Thank you very much for the comments. We think that our main contribution lies in the new multilevel framework of integrating an intermediate stage of layer and track assignment between the coarsening and the uncoarsening stages. As pointed in the paper, this change to the classical multilevel routing framework substantially improve the flexibility and provide a better trade-off in handling many nanometer electrical effects such as crosstalk. (Please also see our response R1A1.) Both MRMCST and CLA/CTA are mainly used to demonstrate the efficiency and effectiveness of our new multilevel routing framework. Further, we have shown in the revision that MRMCST does improve timing, compared to the work [22] which is based on traditional performance-driven routing tree construction. So our MRMCST also contributes to the overall framework.

– **R3Q2:** The coarsening steps are similar to a work by Marek-Sadowska "Global router for gate array" in ICCD 1984. A discussion on similarities, advantages and weakness comparing proposed method is necessary.

– **R3A2:** Thank you very much for the comments. We have added the survey in the revision. Here we give the comparisons:

∗ **Framework comparison:** Although our coarsening stage and the work by Marek-Sadowska[24] both use the hierarchical approach to handle the global routing problem, but the operations are quite different. In the coarsening stage, our global router first finds routing paths for the local nets one by one. After the global routing is performed, we merge 2×2 GCs of G_0 into a larger GC and at the same time perform resource estimation for use at the next level. Coarsening continues until the number of GCs at a level is below a threshold. After the coarsening is finished, a layer/track assignment is performed to assign long and straight segments to underlying routing resources, then the uncoarsening stage tries to refine the routing solution of the unassigned segments. The work in [24] first enumerates possible routing patterns for each net in a 2×2 supercell. With the consideration of boundary capacity and routing cost, the mapping of a set of nets to their routing patterns is determined simultaneously. Then, pattern routing is followed by maze routing, and an artificially intelligent rerouter is applied to do rip-up and reroute.

∗ **Advantages:** In the coarsening stage, we route nets level-by-level according to their wirelength. Short nets are routed first, which tends to have better routability as pointed out in [18] and can update the estimation of routing resources more precisely. Further, our pattern routing approach is much faster than the outer-most loop approach in [24], for which determines routing patterns for all nets and then routes each net by very time-consuming maze routing. As pointed out in [24], the work [24] suffered high time complexity; in contrast, ours leads to a better trade-off in routability, runtime, and scalability.

Note that 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 are usually shorter than those at a higher level. 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 have a higher priority than a longer net as far as routability is concerned. Kastner, Bozorgzadeh, and Sarrafzadeh in[18] also suggest this conclusion. Thought this net ordering scheme may not be the optimal solution for some routing problems, it is still a reasonable alternative.

– **R3Q3:** The MRMCST is similar to BRBC[10], AHHK by Andrew Kahng's group and A-tree by Jason Cong's group. Given so many similar previous works, it is not clear why a new tree construction method is necessary. A theoretic justification or an experimental comparision is a MUST. Why a spanning tree is employed instead of a Steiner tree? an explanation is needed on this as well.

– **R3A3:** Thank you very much for the comments. Our MRMCST tries to find the tree with the minimal critical path while preserving the minimum total wirelength. This formulation is good for routability and performance trade-off because the wirelength not only consumes routing resource (routability), but also incurs capacitance load. To explicitly consider the timing constraint, we can use MRMCST as an initial solution and apply a local improvement technique to fix timing violations. In this revision, for example, we have modified the MRMCST to meet the required arrival time by the recalling-modification method mentioned in the paper to fix timing violations.

We added a new experiment on performance-driven routing in the revision. Compared with [22] that is based on the classical performance-driven routing tree construction (similar to BRBC and AHHK), the experimental results on performance-driven routing (Table 4) have shown that ours outperforms the classical approach by a large margin.

We employed a spanning tree instead of a Steiner tree because its time complexity is low and its routability is reasonably good. Note that the multilevel routing framework is intended for large-scale problems. Therefore, time complexity is an important concern for scalability.

– **R3Q4:** Since in MRMCST no required arrival time is considered, the proposed method is hardly timing driven. MRMCST treats critical nets and non-critical nets equally.

– **R3A4:** Thank you very much for the comments. We have modified the MRMCST to consider the required arrival time by the recalling-modification method to avoid timing violations. Please see Table 4 for the experimental results.

– **R3Q5:** Looks track and layer assigment are performed only at the most sparse grid level. Since short segments have been routed in previous coarsening steps, and only a few long segment are left for the layer and track assignment. Is this adequate to consider only long segment for crosstalk? since those tremendous small segments may affects crosstalk too. Authors need to provide experimental data on how much crosstalk from those long segments, and how much are from short segments.

– **R3A5:** Our multilevel routing framework performs only global routing at the coarsening stage; therefore, short segments are in fact routed after the layer and track assignment. So they can still be considered for layer and track assignment.

Yes, we do agree with you that short segments also contribute to the overall crosstalk. Nevertheless, in our implementation, we performed layer and track assignments only for long segments. The choice is mainly for runtime-complexity reduction and via control (which in turn may affect timing).

For the reviewer's reference, Table 6 shown below gives the ratios of the coupling length of long segments C_L to the total coupling length C_{Total}. As shown in Table 6, more crosstalk comes from long segments.

Circuits	C_L/C_{Total}
S5378	72.91
S9234	74.22
S13207	62.78
S15850	60.35
S38417	59.03
S38584	58.19

Table 6: The ratio of C_L/C_{Total}.

– **R3Q6:** In coarsening steps, pattern routing is applied without justification, again. Pattern routing does not consider wire detour, usually is very poor on congestion removal. It is not clear why it can be used to improve routability.

– **R3A6:** Again, the multilevel routing framework is intended for large-scale problems. Therefore, time complexity is an important concern for scalability. Pattern routing is very fast for routing prediction and is thus suitable for multilevel routing for large-scale designs. Further, the experiments of pattern routing in [17] also showed reasonably good routability.

– **R3Q7:** In problem formulation of CLA and CTA, the cost are not defined. Cost between wire segments are defined as adjacency length, but cost between a segment and a layer (or a track) has never been defined.

– **R3A7:** Thank you very much for the comments. We have rephrased the problem descriptions in the revision. The costs are both based on the coupling length. For CLA, given a set of layers L and a set of segments ℓ, the problem finds an assignment of segments to the layers such that the sum of the overlapping lengths among all nets in every layer is minimized. That is, the cost for CTA simply comes from the overlapping lengths of nets since nets are not yet assigned to tracks during the layer assignment and all information we have is the spans of nets.

– **R3Q8:** In experimental results, delay data are provided. Please specify the delay model. Coupling capacitance is not considered in the delay estimation, and authors claim that a better result on delay can be reached if coupling cap is included. This claim is roughly true, but not always, since same amount of coupling cap may generate different impact on delay depending on the location of the coupling caps.

– **R3A8:** Thank you very much. We use the Elmore delay model for delay computation (added in the revision). We have also rephrased our comments on adding coupling capacitance for delay computation.

– **R3Q9:** The rerouting in uncoarsening steps may destroy the crosstalk driven layer/track assignment results. Not clear if crosstalk is considered in the uncoarsening steps.

– **R3A9:** In the uncoarsening stage, routability is typically the most important concern in our multilevel routing framework (since the remaining nets are much "tougher" for routing). Furthermore, there is usually not much room for crosstalk improvement in this stage because just a few nets are rerouted and most nets are fixed.

– **R3Q10:** In experimental results, it shows that the proposed method wins over [21] on almost every metric: CPU, delay, routability, crosstalk. For the crosstalk part, it is obviously. But it is not obvious why the proposed method can do better job on all of CPU, delay and routability than [21]. Some explanations are needed here. Just throwing an algorithm and a result without explanation is not sufficient, particularly when the problem is very complex.

– **R3A10:** Thank you very much for the comments. We have added the following explanation in the revision: "The results reveal the effectiveness of the intermediate stage of layer and track assignments and our MRMCST for performance-driven routing tree construction. Since many segments are routed in the layer/track assignment stage (which is very efficient), the search space during the uncoarsening stage is significantly reduced. Consequently, the running time and solution quality can be improved simultaneously. Also, compared with [5], [22] that are based on the classical performance-driven routing tree construction, the experimental results on timing have shown that our MRMCST leads to significantly better maximum and average delays."
Note that these experiments are based on comparable routability to [22] ([21] in the original version). So we claim only the improvements for delay, crosstalk, and running time.