A Fast Crosstalk- and Performance-Driven Multilevel Routing System

Tsung-Yi Ho[1], *Yao-Wen Chang*[2]*, *Sao-Jie Chen*[2]†, *and D. T. Lee*[3]
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan[1]
Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan[2]
Institute of Information Science, Academia Sinica, Taipei, Taiwan[3]

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, the experimental results show that our approach 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.

1 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 [18] and line-searching approaches [14], which route net-by-net. Most concurrent algorithms apply network-flow [1] or linear-assignment formulation [5, 22] to route a set of nets at one time.

The major problem of the flat framework 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 [22] proposed a hierarchical global router based on linear assignment. Chang, Zhu, and Wong [5] 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 [7], 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, 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* [15], the multilevel placer, *mPL* [4], and the multilevel floorplanner/placer, *MB*-tree* [19], 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 [13, 21]. Lin, Hsu, and Tsai in [21] and Hayashi and Tsukiyama in [13] presented hybrid hierarchical *global* routers for multi-layer VLSI's [13], in which both bottom-up (coarsening) and top-down (uncoarsening) techniques were used for global routing. Recently, Cong, Fang, and Zhang proposed a pioneering multilevel global-routing approach for large-scale, full-chip, routability-driven routing [7]. Cong, Xie, and Zhang later proposed an enhanced multilevel routing system, named MARS [8], 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 [7]. 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 [20]. 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.
- A point-to-path maze-searching algorithm is proposed for better wirelength and routability optimization.
- An efficient and effective layer/track assignment scheme is incorporated for crosstalk and run-time optimization.

Figure 1 shows our multilevel framework. 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

*Yao-Wen Chang's work was partially supported by the National Science Council of Taiwan ROC under Grant No. NSC 91-2215-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.

ICCAD'03, November 11-13, 2003, San Jose, California, USA.

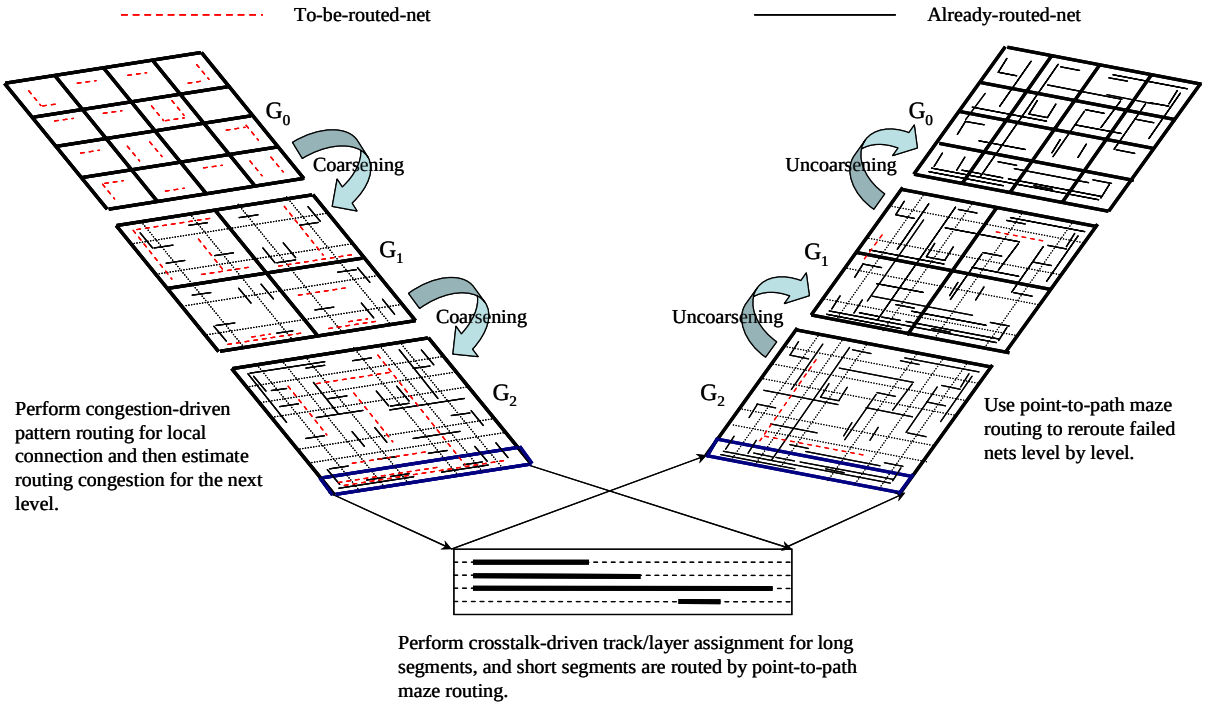


Figure 1: Multilevel framework flow

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 and rip-up and re-route to refine the routing solution level by level.

Compared with [20], the experimental results show that our approach 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. The results show the promise of our approach.

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

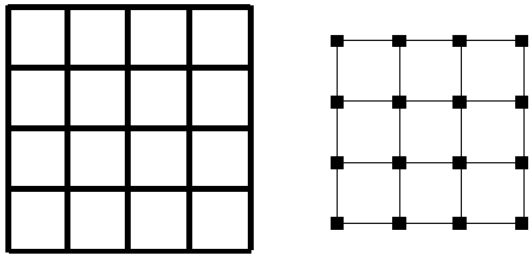
2 Preliminaries

2.1 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 ID. 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 [12]. Wires in each layer run either horizontally or vertically. We refer the layer as a horizontal (H) or a vertical (V) routing layer.



(a) Partitioned layout (b) Routing graph

Figure 2: The routing graph.

2.2 Multilevel Routing Model

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

3 Multilevel Routing Framework

Our multilevel routing algorithm is inspired by the work [20]. Nevertheless, our framework is significantly different from that of [20]. Different from the framework of [20] 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 [16] 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.

3.1 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 ([10]). 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 [10], 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 [10], 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 [23], 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. 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 [26], 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.

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 $\varepsilon_1, \varepsilon_2, ..\varepsilon_{max}$ 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 ε_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

Figure 3: Algorithm for constructing an MSTUG and an MSTIG.

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:

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

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

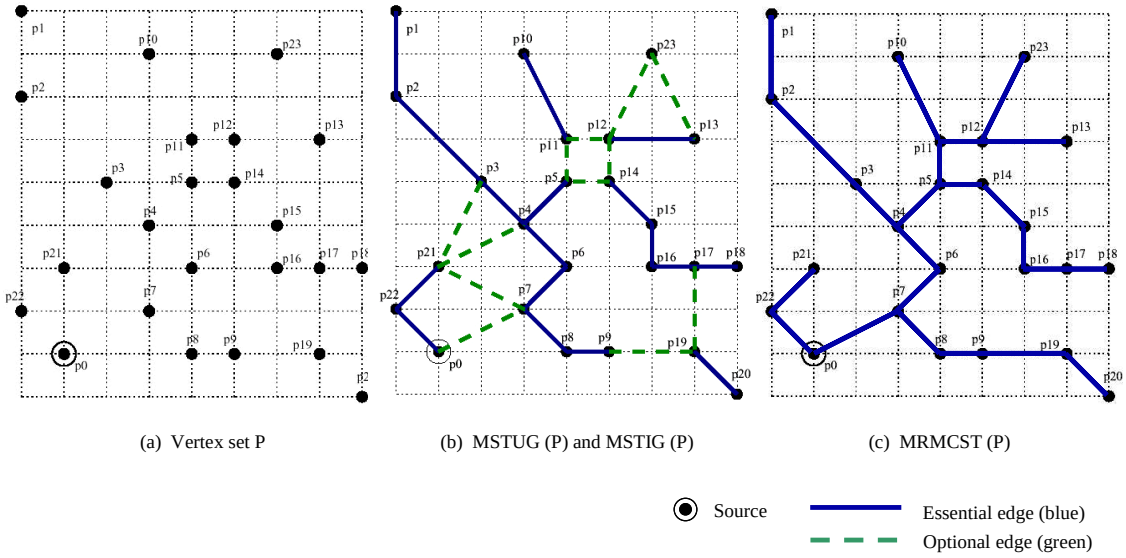
The MRMCST algorithm that employs LOCS is summarized in Figure 5. See Figure 4(c) for the MRMCST of Figure 4(b).

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.*

3.2 Crosstalk-Driven Layer/Track Assignment

As fabrication technology shrinks into the VDSM era, as pointed out in [25], 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 assume that there is crosstalk only between adjacent parallel wires.



(a) Vertex set P (b) MSTUG (P) and MSTIG (P) (c) MRMCST (P)

Figure 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.

Recently, there has been much research on the coupling problem in both global and detailed routing. Zhou and Wong [27] minimized crosstalk at the global routing stage. Chaudhary et al. [6] 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 trying to embed a late-routing net that must traverses a region already dense with conflicting aggressor or victim nets.

To address these problems, Kay and Rutenbar [17] 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. Baterywala et al. [3] proposed a fast track assignment heuristic considering routability, but crosstalk was not addressed in the work.

In this paper, we propose a fast layer/track assignment heuristic for crosstalk optimization. After the coarsening stage, we may obtain some 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, a set of segments ℓ, and a cost function $\Gamma : \ell \times L \longrightarrow N$ which represents the coupling cost of assigning a segment to a layer, find an assignment that minimizes the sum of the coupling costs of each layer.

The CLA problem can be formulated as the max-cut, k-coloring (MC) problem [24]. However, the MC problem is NP-complete [24]. 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 T be the set of tracks inside a panel. Each track $t \in T$ 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 $t \in T$, $t \equiv \biguplus x_i$, 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. Thus, the crosstalk-driven track assignment problem can be defined as:

- **The Crosstalk-driven Track Assignment (CTA) Problem:** Given a set of tracks T, a set of segments ℓ, and a cost function $\Phi : \ell \times T \longrightarrow N$ which represents the coupling cost of assigning a segment to a track, find an assignment that minimizes the sum of the coupling costs of the assignment.

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 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).

The CTA problem can be formulate as the Hamiltonian path problem which has been proven to be NP-complete [11]. 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 graph is an

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 B_i)
17 Update $dist(w, c)$ and mark "visited" while traversing B_i from v_2;
18 For each "uninserted" optional edge $e = (v_3, v_4)$, $v_3 \in B_i$ and $v_4 \in B_j$
19 FirstCost = $cost(v_3, v_4)$;
20 SecondCost = $dist(s, v_3) + cost(v_3, v_4) + dist(v_4, pc(B_j)) + ecc(pc(B_j))$;
21 InsHeap(e, FirstCost, SecondCost);
22 Mark e "inserted";
23 MRMCST = T_s;
end

Figure 5: Algorithm for constructing an MRMCST.

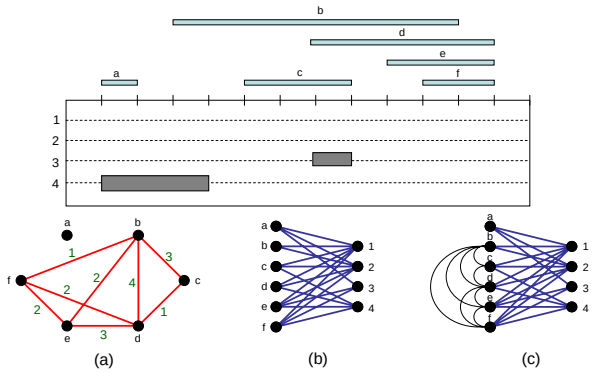


Figure 6: Constraint graph modeling for track assignment. (a) The subHCG for the given instance. (b) The corresponding bipartite assignment graph. (c) The combination graph.

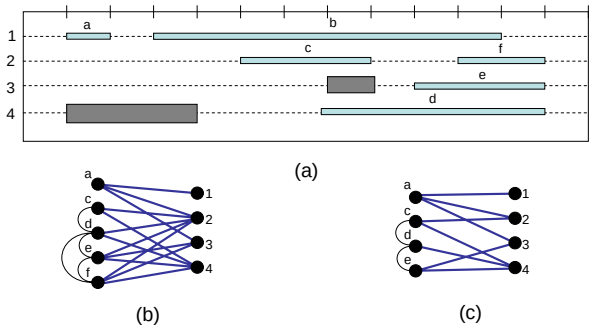


Figure 7: The process for track assignment. (a) The final track assignment for the instance of Figure 6. (b) The resulting combination graph after assigning b to track 1. (c) The resulting combination graph after assigning f to track 2.

interval graph, 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.

4 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 [20] based on the six benchmark circuits provided by the authors. See Table 1 for the benchmark circuits. (Note that the benchmark circuits used in [20] also contain Mcc1, Mcc2, Struct, Prim1 and Prim2. However, as pointed out in [20], 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 1.) The design rules for wire/via widths and wire/via separation for detailed routing are the same as those used in [20].

Table 1 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 [20] are better than those in [9] and [7], we shall compare our multilevel router with that in [20].

Experimental results on run-time, routing completion rate, delay, and crosstalk are listed in Tables 2 and 3, where "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. To perform experiments on timing-driven routing, we used the same resistance and capacitance parameters as those used in [20] and set the constraint ratio k used in [20] to 5.5 for comparison. (For this case, both routers have comparable routability, and thus it is easier to compare the delay and crosstalk results.) A via is modeled as the Π-model circuit, with its resistance and capacitance being twice of those of a wire segment. All the parameters were the same as those

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 1 : The benchmark circuits.

Circuits	Results of [20]				Our Results			
	Time (s)	Memory (MB)	#Failed nets	Cmp. Rates	Time (s)	Memory (MB)	#Failed nets	Cmp. Rates
S5378	35	18	5	99.7%	10.6	34	5	99.8%
S9234	26.2	14	4	99.7%	8.1	33	2	99.9%
S13207	106.7	24	7	99.8%	22.6	53	10	99.8%
S15850	538.8	40	31	99.3%	62.6	79	19	99.7%
S38417	899.9	75	57	99.5%	71.3	160	33	99.8%
S38584	1953.7	496	59	99.6%	255.6	977	54	99.8%
Comp.	6.7	1	1.4	1	1	2.1	1	1

Table 2 : Results of run-time and routability comparison.

Circuits	Results of [20]				Our Results			
	D_{max}	D_{avg}	C_{max}	C_{avg}	D_{max}	D_{avg}	C_{max}	C_{avg}
S5378	37308	1403	507	25.4	27577	1258	342	20.2
S9234	25512	1072	579	21.7	23591	1009	426	17.8
S13207	55337	1262	1526	29.2	52034	1243	1211	22.7
S15850	76297	1302	2913	28.3	68317	1253	2274	22.9
S38417	121419	1170	5704	25.6	105575	1146	4732	20.9
S38584	150936	1208	23196	26.8	131877	1151	18810	22.6
Comp.	1.2	1.1	1.3	1.2	1	1	1	1

Table 3 : Results of timing and crosstalk comparison.

used in [20], and both routers were run on the same machine. Compared with [20], 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.

It should be noted that the coupling capacitance is not included in delay computation for fair comparison with [20]. If coupling capacitance is considered, we can obtain even better timing reduction due to the significant crosstalk reduction.

5 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 nanometer electrical effects.

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, J. R. Shinnerl, "Multilevel optimization for large-scale circuit placement," *Proc. ICCAD*, pp. 171–176, Nov. 2000.

[5] 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.

[6] K. Chaudhary, A Onozawa, and E. S. Kuh, "A spacing algorithm for performance and crosstalk reduction ," *Proc. ICCAD*, pp. 697–702, Nov. 1993.

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

[8] J. Cong, M. Xie and Y. Zhang, "An enhanced multilevel routing system," *Proc. ICCAD*, pp. 51-58, Nov. 2002.

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

[10] 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.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms," *MIT Press*, 2001

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.

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

[19] 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.

[20] S.-P. Lin and Y.-W. Chang, "A novel framework for multilevel routing considering routability and performance," *Proc. ICCAD*, pp. 44-50, Nov. 2002.

[21] 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.

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

[23] 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.

[24] 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.

[25] D. Sylvester et al, "Interconnect scaling: Signal integrity and performance in future high-speed CMOS designs," *Proc. VLSI Symposium on Technology*, 1998.

[26] R. E. Tarjan, "Data structures and network algorithms," *CMBS 44*, SIAM, 1983.

[27] H. Zhou and D. F. Wong, "Global routing with crosstalk constraints," *Proc. DAC*, pp. 374-377, Jun. 1998.