# A Gridless Routing System with Nonslicing Floorplanning-Based Crosstalk Reduction on Gridless Track Assignment

YIH-LANG LI, YU-NING CHANG, and WEN-NAI CHENG, National Chiao-Tung University

Track assignment, which is an intermediate stage between global routing and detailed routing, provides a good platform for promoting performance, and for imposing additional constraints during routing, such as crosstalk. Gridless track assignment (GTA) has not been addressed in public literature. This work develops a gridless routing system integrating a congestion-driven global router, crosstalk-driven GTA and an enhanced implicit connection-graph-based router. Initial assignment is produced rapidly with a left-edge like algorithm. Crosstalk reduction on the assignment is then transformed to a restricted nonslicing floorplanning problem, and a deterministic O-Tree based algorithm is employed to reassign each net segment. Finally, each panel is partitioned into several subpanels, and the subpanels are reordered using branch and bound algorithm to decrease the crosstalk further. Before detailed routing, routing tree construction is undertaken for placed IRoutes and other pins; many original point-to-point routings are set to connect to IRoutes, and can be accomplished simply with pattern routing. For detailed routing, this work proposes a rapid extraction method for pseudomaximum stripped tiles to boost path propagation. Experimental results demonstrate that the proposed gridless routing system has over 2.02 times the runtime speedup in average for fixed- and variable-rule routings of an implicit connection-graph-based router, NEMO. As compared with a commercial routing tool, this work yields an average reduction rate of 13.8% in coupling capacitance calculated using its built-in coupling capacitance estimator.

Categories and Subject Descriptors: B.7.2 [**Integrated Circuits**]: Design Aids—*Placement and Routing*

General Terms: Algorithms, Design, Performance, Reliability

Additional Key Words and Phrases: Gridless routing, non-slicing floorplanning, crosstalk reduction, implicit connection-graph based router, full-chip routing, detailed routing

**ACM Reference Format:**
Li, Y.-L., Chang, Y.-N., and Cheng, W.-N. 2011. A gridless routing system with nonslicing floorplanning-based crosstalk reduction on gridless track assignment. ACM Trans. Des. Autom. Electron. Syst. 16, 2, Article 19 (March 2011), 25 pages.
DOI = 10.1145/1929943.1929951 http://doi.acm.org/10.1145/1929943.1929951

## 1. INTRODUCTION

Crosstalk effect brings a serious challenge for successfully designing a reliable and fast Very Large Scale Integrated (VLSI) circuit. Continuous progress in semiconductor technology with shrinkage in feature size raises coupling capacitance and worsens this phenomenon. Crosstalk minimization can be realized in different circuit levels. This work focuses on the crosstalk minimization during interconnection design in physical level.

This work was supported in part by the National Science Council of Taiwan under Grant NSC 98-2220-E-009-059, in part by Grant NSC 97-2220-E-009-036, in part by Grant NSC 97-2220-E-009-002, and in part by Grant NSC 98-2220-E-009-025.
Authors' addresses: Y.-L. Li, Y.-N. Chang, and W.-N. Cheng, Department of Computer Science, National Chiao-Tung University, Hsin-Chu, Taiwan; email: ylli@cs.nctu.edu.tw.

DOI 10.1145/1929943.1929951 http://doi.acm.org/10.1145/1929943.1929951

### 1.1. VLSI Routing Flow

Conventional routing flow comprises global and detailed routing [Sait and Youssef 1999]. Global routing typically partitions the entire routing region into subregions, that is, *global cells* (GCells). Every subregion has a capacity denoting the maximum number of nets that can pass through it. A global router creates a loose route from a subset of global cells connecting the terminals of the net without specifying the actual geometric layout wires. Detailed routing discovers the actual wire segment for every net within the assigned GCells from the global router. Since a detailed router generally only connects the pins with fixed positions, the cross-point of a net on the boundary of two linked GCells has to be determined. Kao and Parng [1995] proposed a heuristic algorithm to optimize the pin alignment across the boundaries of successive GCells. As the increase in design complexity lasts, the detailed routing consumes increasing runtime in processing modern designs, even with the aid of global routing. Cong et al. [2001] presented a three-stage routing system, comprising performance-driven global routing, congestion-driven wire planning considering variable-rule wires and gridless detailed routing. Wire planning is first applied to all nets one by one before detailed routing to determine the passing regions for each net. Wire planning also helps an incomplete net to refine its new passing regions, by viewing previously completed wires as new obstacles.

Wire planning only determines passing regions of a net instead of its placed track, and processes one net at a time. After that, the track assignment (TA), that is, an intermediate stage between global routing and detailed routing, is incorporated into the two-stage routing flow to lower the routing runtime, and to produce more straight wires than Lee's algorithm [Batterywala et al. 2002]. TA only handles nets called *IRoutes* that completely pass through at least one global cell, and places as many nets as possible on available tracks. IRoute *overlap graph* (OLG) is practically a horizontal constraint graph, and is initially constructed to determine whether two IRoutes can be assigned in a track. The maximum clique in an OLG is identified in order to assign them to different tracks. The assignability of each IRoute to each track is thus denoted by a bipartite graph based on the pre-placed blockages on each track. The TA of the current clique can then be solved using a weighted bipartite matching algorithm. IRoutes are processed clique by clique. TA thus completes the routing of most long nets, and significantly simplifies detailed routing. Before the work of Batterywala et al. [2002], the concept of TA was applied in the constrained via minimization to better a completed routing design [Kuo et al. 1988; Shi 1997; Chang and Cong 1999].

### 1.2. Routing Model

The grid-based routing model, in which a routing region is split into a uniform-size grid array, is generally adopted to solve most routing problems. Since the grid-based model has a simple data structure, it typically constructs routing models quickly, and allows fast neighbor identification and one-step move. For the deep submicron (DSM) and namometer designs, a variable rules for routing wires is stipulated to decrease the crosstalk and wire delay, thus improving reliability and performance. The grid-based routing model can also be extended to solve variable-rule routing problems, while consuming additional routing resources. The gridless routing model is more flexible but also more complicated than the grid-based model, and is proposed to avoid the need for additional routing resources to accommodate variable routing rules in the routing graph.

Two well-known gridless routers are tile-based and implicit connection-graph-based routers, which possess the advantages of low path propagation complexity and fast routing graph construction, respectively [Margarino et al. 1987; Dion and Monier 1995;

Cong et al. 2001]. Tile-based routers partition the routing area into a maximum horizontally or vertically stripped tile plane by extending each horizontal or vertical border line of existing obstacles until it reaches another obstacle or the boundary of the routing region. Tile propagation is performed on the tile plane with the *postponed arbitrary choice* scheme, which only identifies a list of passed tiles from the routing path, rather than discovering the precise path positions. Finally, the path construction algorithm identifies a minimum-cost path in the tile list. Implicit connection-graph-based routers extend the border lines of all obstacles to penetrate other obstacles until they reach the boundaries of the routing region. A maze routing algorithm is thus undertaken on the non-uniform grid plane to discover a feasible path. Li et al. [2007] presented a novel gridless router with the benefits of tile-based and implicit connection-graph-based routers, called NEMO. By considering the implicit connection graph as a tile array instead a grid array, NEMO quickly builds the routing graph with point array, while adjacent and equal-height or width tiles are grouped as a *pseudo maximum horizontally or vertically stripped tile* (*PMT*) to acquire fast tile propagation. A PMT is dynamically identified when tile propagation explores an unvisited space region, that is, tile propagation in NEMO includes PMT extraction and propagation. An analysis of routing runtime statistics demonstrates that tile propagation consumes most of the routing runtime, so simplifying PMT extraction is an effective way to accelerate tile propagation in NEMO.

### 1.3. Crosstalk Optimization in Routing

Interconnection complexity rises as the features become smaller. Therefore, the noise between wires needs to be eliminated. Crosstalk minimization has been investigated in every stage of routing flow. Xue et al. [1996] originally developed a method for decreasing the estimated crosstalk risk at the global routing stage. The crosstalk optimization process involves three components, crosstalk risk estimation (CRE), risk tolerance bound partitioning (RTBP) and global route adjustment (GRD). CRE constructs a crosstalk risk graph to record the status of crosstalk risk for every GCell. RTBP attempts to remove the high-risk GCell by partitioning the crosstalk bound, and performing GRD if high-risk nets still exist following RTBP. Every net is removed, and its risk reduction is examined. The nets with maximum risk reduction are selected for rip-up and rerouting to identify a minimum-cost path. Zhou and Wong [1999] also presented a global router that addresses the crosstalk impact. Their router identifies a minimum-crosstalk Steiner tree using a Steiner-tree heuristic algorithm based on the shortest path and the minimum spanning tree, then performs a Lagrangian relaxation based rip-up and rerouting scheme on the nets with crosstalk violation.

To manage cross-point assignment, Chang and Cong [2001] presented a two-stage assignment method (coarse pseudopin assignment and detailed pseudopin assignment) to control the crosstalk noise while minimizing the cost of wire length and the number of vias [Chang and Cong 2001]. Tseng et al. [2001] also proposed a model that minimizes crosstalk at the cross-point assignment stage, and employs timing-driven cross-point assignment flow. Their approach applies net ordering to the most timing-critical channel, followed by a space relaxation algorithm to reduce the coupling capacitance of the most timing-critical segment. Crosstalk reduction during layer assignment is addressed in Cho et al. [1993]. Ho et al. [2005] developed a crosstalk-driven TA in a multilevel routing system, decreasing crosstalk by identifying a minimum-cost (IRoute overlap length) Hamiltonian path of the currently processed clique obtained from the IRoute OLG. Wu et al. [2005] proposed a timing-driven TA algorithm that maximizes the minimum timing slack among all nets by considering the impact of coupling capacitance and wire detour on timing delay.

Some works have addressed crosstalk optimization in channel routing [Gao and Liu 1996; Vittal and Marek-Sadowska 1997; Sapatnekar 2000]. Gao and Liu [1996] presented a mixed ILP formulation to optimize the crosstalk in the gridded channel routing problem. They employed horizontal segment permutation to transform a channel routing result produced by an existing channel routing algorithm to another crosstalk-reduced routing result with the same channel height. The permutation range of every horizontal segment is bounded by its vertical constraints. Alteration of a horizontal segment affects the length of its coupling with neighboring nets, and of its vertical connection to its pin, thus influencing the coupling length induced by the vertical segment. Sapatnekar [2000] proposed a polynomial-time algorithm in the worst case to efficiently measure the impact of crosstalk on the delay of a net. The method is then applied to decrease the crosstalk in channel routing, again with wire permutation. Vittal and Marek-Sadowska [1997] first adopted the traveling salesperson algorithm to yield a minimum-crosstalk net ordering to enter the processed channel. The greedy channel algorithm then solves the channel routing problem, and further decreases the crosstalk by inserting jogs after the split-net collapsing step.

### 1.4. Contributions

Although previous works have developed a complete three-stage grid-based crosstalk-driven routing flow, the research on gridless routing flow is still incomplete. Gridless TA and crosstalk optimization in gridless TA have not previously been investigated. This work firstly develops an efficient crosstalk-driven gridless routing system, comprising a congestion-driven global router, a crosstalk-driven gridless TA (GTA) and enhanced NEMO with fast PMT extraction. The GTA in this work has two phases, initial GTA and crosstalk reduction. In the first phase, a left-edge like algorithm is invoked to obtain an initial TA result quickly. The problem of reassigning IRoutes for crosstalk optimization is then transformed into a restricted nonslicing floorplanning problem. A fast deterministic floorplanning algorithm is employed to replace each IRoute in the position with most decreased crosstalk. After refining the position for each IRoute, every panel is treated as a collection of subpanels, each of which can also be considered as a hybrid IRoute with different states on its top and bottom borders for a horizontal IRoute. Reordering the subpanel can further decrease the total overlapping length. Before detailed routing, routing tree construction is undertaken for placed IRoutes and other pins; many original point-to-point routings are set to connect to IRoutes, and can be completed simply with pattern routing. For detailed routing, this work presents a rapid PMT extraction method to boost path propagation. Experimental results demonstrate that the proposed gridless routing system can perform over 2.02 times faster for fixed- and variable-rule routings than an implicit connection-graph-based router, NEMO. As compared with a commercial routing tool, this work yields an average runtime speedup of 8.16 times and an average 13.8% reduction rate in coupling capacitance calculated with the built-in coupling capacitance estimator of the commercial routing tool.

The rest of this article is organized as follows. Section 2 briefly reviews NEMO, GTA, crosstalk model, and the proposed routing system. Section 3 presents crosstalk-driven GTA. Section 4 presents the detailed routing. Section 5 discusses the routing results using two rule sets. Finally, Section 6 draws conclusions.

## 2. PRELIMINARIES

### 2.1. NEMO Overview

The implicit connection graph-based router and tile-based router are conventionally adopted gridless routers. The former conducts routing on a non-uniform grid

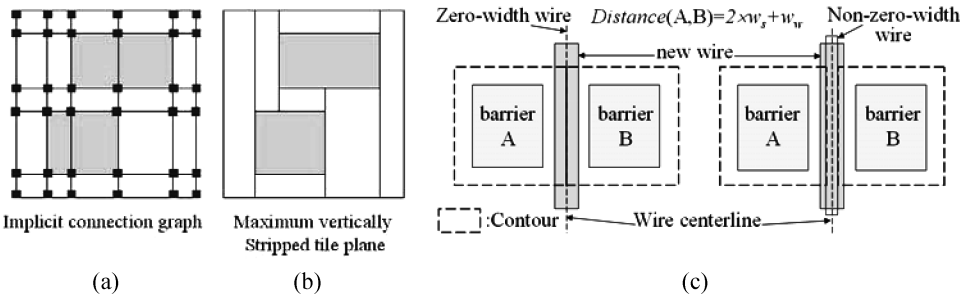



Fig. 1. (a) Implicit connection graph is represented by a nonuniform grid array; (b) the maximum vertically stripped tile plane for the same problem in (a); (c) zero-width wire model and non-zero-width wire model.

array (Figure 1(a)) and the latter conducts routing on a corner-stitching tile plane (Figure 1(b)). Figure 1(b) presents a *maximum vertically stripped* (*MVS*) tile plane that is obtained by extending all vertical borders of barriers until a region boundary or another barrier's border is reached. A *maximum horizontally stripped* (*MHS*) tile plane can be generated similarly. The example in Figure 1(c) elucidates the primary difference between the wire models (zero-width and non-zero-width wire models) of these two gridless routers. Two barriers in the example are spaced $2 \times w_s + w_w$ apart, where $w_s$ and $w_w$ are the minimum wire separation and minimum width. The contours of the two barriers will abut on one side in the zero-width wire model (the left part in Figure 1(c)), whereas the two contours whose values are slightly smaller than those of the former contours, such as by one unit of width, are separated by a distance of double this value in the non-zero-width wire model (right part in Figure 1(c)). A traditional implicit connection graph-based router is superior to the tile-based router in the construction of a routing graph (grid operation vs. tile operation), while the tile-based router outperforms the implicit connection graph based router in path searching (tile expansion vs. grid maze). NEMO employs the non-zero-width wire model since NEMO views a nonuniform grid array as a group of tiles, each of which is represented by its left lower grid point. In this work, $L_{lx}(i)/L_{ly}(i)$ denotes the $i$-th vertical/horizontal grid line in the grid array on layer $l$. NEMO undertakes routing on an implicit connection plane in the same way as tile expansion on a corner-stitching tile plane. During path propagation, NEMO groups a series of adjacent free tiles as a PMT to enable path searching. The definition of PMT is as follows.

*Definition PMT*. A *PMT* on an implicit connection plane of a horizontal/vertical routing layer comprises maximally connected free tiles in a ROW/column.

If two adjacent PMTs have the same left and right top and bottom borders, then they merge into a new PMT. Every identified PMT is equivalent to a maximum horizontally/vertically stripped free tile. For instance, the implicit connection graph in Figure 2(a) constitutes a $7 \times 6$ grid array. Cong et al. [2001] identified a routing path along the grid lines. However, NEMO regards the $7 \times 6$ grid array as a $6 \times 5$ tile array and every tile is signified by the coordinate of its left lower corner. To validate the legality of every tile in the first time to visit, a slit and interval tree stores all existing blockages, providing an efficient query scheme. If a tile is verified as free, then adjacent free tiles are grouped as a PMT by investigating the slit and interval tree leftwards and rightwards along the leaf nodes and upwards, to search for the blockages stored in unvisited nodes on the upper level. For instance, in Figure 2(a), a PMT extraction request is invoked at point $Q$ on a horizontal-layer implicit connection graph with four blockages.

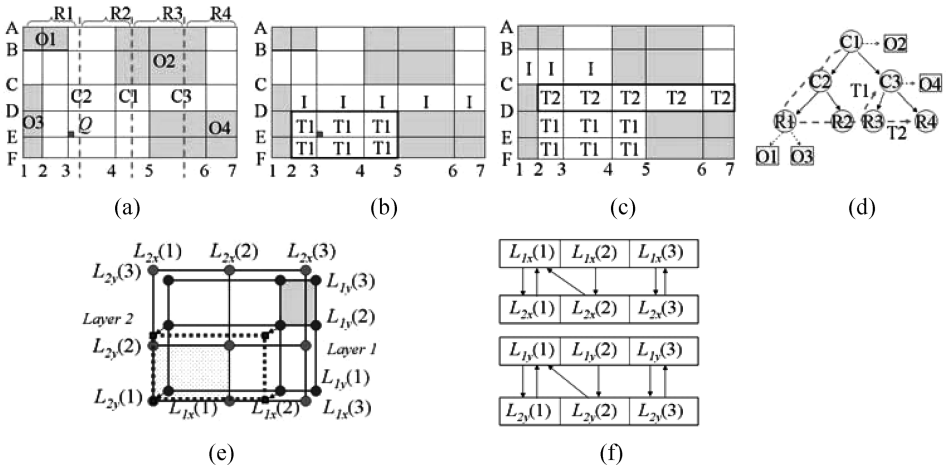



Fig. 2. (a) A PMT extraction request is invoked at the point $Q$ on a horizontal-layer implicit connection graph containing four blockages; (b) PMT $T1$ and another incomplete PMT marked by $I$ are identified; (c) PMT $T2$ and another incomplete PMT $I$ are identified; (d) bold dotted lines indicates traverse steps on the slit and interval tree of routing region (a) for identifying PMTs $T1$ and $T2$; (e) an example of layer switching from a tile on *Layer* 1 to tiles on *Layer* 2; (f) an example of two adjacent *CAs* and their related *PAs*.

The related slit and interval tree is built through three interval cut lines, as illustrated in Figures 2(a) and 2(d), where the routing region is partitioned into four sub-regions. A query at point $(3, E)$ yields a PMT from $(2, E)$ to $(5, D)$. Successive queries at points $(3, F)$ and $(3, D)$ generate two PMTs, from $(2, F)$ to $(5, E)$ and from $(2, D)$ to $(7, C)$, respectively, where the former merges with the first found PMT as a PMT *T1*, and the latter becomes an incomplete PMT for subsequent usage, as shown in Figures 2(b) and 2(c). Further path propagation demands PMT expansion, and identifies a second PMT *T2* and incomplete PMT starting from $(1,C)$ to $(4,B)$, as shown in Figure 2(c). Figure 2(d) displays the tree traversal processes for PMTs *T1* and *T2*. The legality validation and PMT extraction are conducted whenever the router intends to explore a neighboring unvisited region. Since the number of such explorations before reaching the target is very large, the time spent in identifying PMT is a significant issue.

On the issue of layer switching, NEMO employs a constant-time layer-switching mechanism to hasten path searching: a coordinate array (*CA*), and a projection array (*PA*). Each *CA* represents the left bottom corner of a tile and each *PA* signifies the tiles in adjacent layers that can be firstly reached from one tile. The formal definition is as follows; the *i-th* element of a *CA* for layer $l$, say $L_{lx}(i)$, has a related element in the *PA*, which points to the *j-th* element in the *CA* of the adjacent layer, say $l + 1$, when $L_{(l+1)x}(j) \leq L_{lx}(i) < L_{(l+1)x}(j + 1)$. The *i-th* horizontal element of the *PA* from layer $l$ to $l + 1$ is denoted $PA_{(l,l+1)x}(i)$. For example, Figure 2(e) displays two-layer routing planes with a single barrier in each plane, while Figure 2(f) depicts their related *CA* and *PA* arrays. The intended tile propagation in Figure 2(e) is from a layer-one tile $T(L_{1x}(1), L_{1y}(1))$ to layer two, and the query about reachable tiles on layer two is answered by accessing the *PA* array to determine the region outlined in bold dotted borders in layer two onto which the tile $T(L_{1x}(1), L_{1y}(1))$ on layer 1 is projected in constant time. All free tiles on layer two that cover the projection region are reachable free tiles on layer two through the tile $T(L_{1x}(1), L_{1y}(1))$. Figure 2(f) presents the assignment of the *PA*s in horizontal and vertical directions in layers one and two. For instance, $L_{2x}(2) < L_{1x}(2) < L_{2x}(3)$ and $L_{1x}(1) < L_{2x}(2) < L_{1x}(2)$, so $PA_{(1,2)x}(2)$ equals 2 and $PA_{(2,1)x}(2)$ equals 1. To identify the tiles on layer two accessible from

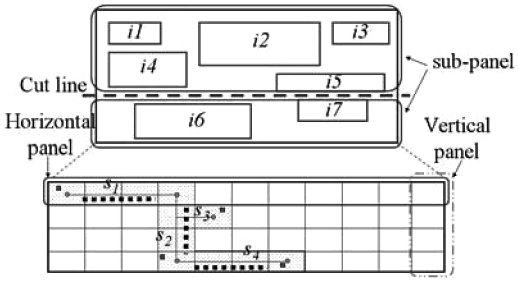



Fig. 3. A routing region is partitioned into an $11 \times 4$ GCell array; the assignment of the topmost panel with seven IRoutes is zoomed in on the top, and can be divided into two subpanels with a cut line.

tile $T(L_{1x}(1), L_{1y}(1))$, $PA_{(1,2)x}(1)$ and $PA_{(1,2)x}(2)$ yield the horizontal range of reachable tiles on layer two, starting from $L_{2x}(1)$, and going toward $L_{2x}(2)$. Similarly, the vertical range of reachable tiles on layer two, from $L_{2y}(1)$ to $L_{2y}(2)$, can be rapidly identified using $PA_{(1,2)y}(1)$ and $PA_{(1,2)y}(2)$. Finally, the tiles located within the area enclosed by the horizontal and vertical ranges of the reachable tiles are the tiles that overlap $T(L_{1x}(1), L_{1y}(1))$. In this case, tiles $T(L_{2x}(1), L_{2y}(1))$, $T(L_{2x}(1), L_{2y}(2))$, $T(L_{2x}(2), L_{2y}(1))$, and $T(L_{2x}(2), L_{2y}(2))$ are candidates for layer switching.

Other approaches for encouraging detailed routing in NEMO are gridline reduction and pseudoblockage insertion. Gridline reduction produces a simplified connection plane by disregarding the gridlines induced from those blockages entirely out of the global path found by the global router. Pseudoblockage insertion places fictitious blockages around the global path to cease PMT extraction when the interval tree query reaches the global path boundary.

### 2.2. GTA and Crosstalk Model

Global router typically partitions a routing region into a GCell array. Figure 3 depicts a routing example of $11 \times 4$ GCells at its bottom of the figure. This section uses the following definitions.

*Definition Panel*. A complete row or column in a GCell array is regarded as a *panel*.

*Definition IRoute*. An *IRoute* is a straight global connection that entirely passes through at least one global cell.

*Definition Subpanel*. If the assignment of all IRoutes within a panel can be partitioned into several disjoint sets using cut lines, then a panel is thereby partitioned into several *subpanels* by the cut lines. A cut line is a line that does not intersect any IRoute, and can partition IRoutes into two disjoint sets.

*Definition Overlap Graph (OLG)*. An *OLG* $G(V,E)$ is a undirected graph, where every vertex in $V$ represents an IRoute and two vertices are connected by an edge in $E$ if their related IRoutes overlap.

The dotted GCells in Figure 3 show the global routing results for a four-pin net, where rectangles are pins and circles are the centers of the GCells. The routing tree can be decomposed into four segments, $s_1$, $s_2$, $s_3$, and $s_4$, yielding three IRoutes, which are represented by bold dotted lines in Figure 3. Tree segment $s_4$ is not processed in the TA stage since it does not completely pass through a GCell. Notably, in this work, every IRoute ends at the GCells next to the GCells at the end of every tree segment. For

instance, in Figure 3, segment $s_1$ is four-GCell long, but only yields a two-GCell-long IRoute (bold dotted lines). The routing that connects the ending GCells to an IRoute is completed by coming detailed routing. Accordingly, the length of every IRoute remains fixed throughout the TA stage. For example, two horizontal IRoutes in Figure 3 do not shrink or expand as the vertical IRoute slides within its panel. The horizontal and vertical wires that are produced in detailed routing and used to connect segments $s_1$ and $s_4$ to segment $s_2$ will vary as the IRoutes of segments $s_1$, $s_2$, and $s_4$ are placed at different tracks. Since the length variation is less than one GCell's height or width, the crosstalk variation caused by this length variation is ignored in this work. The topmost horizontal panel contains 7 IRoutes. GTA has no track to be used. One assignment of the topmost horizontal panel is displayed on the top of the figure. The assignment induces one cut line, given by the bold dotted line in Figure 3, to split the panel into two subpanels. The cut line in Figure 3 splits seven IRoutes into two sets, namely five IRoutes for one set and two for the other.

Observations in previous investigations [Tu et al. 2003; He and Xu 1999] reveal that two highly coupling-capacitance-related factors are the space and overlapping length between two wires. The impact of wire width has been observed not to be important for variable-width routing [He and Xu 1999], indicating that the wire width has only a small influence on coupling effects, with from doubling or tripling the wire width producing a coupling effect variation of about 0.4% to 7%. Consequently, the coupling effect estimation ignores the wire width, and this study adopts the coupling capacitance model used in Zhou and Wong [1999], Tseng et al. [2001], and Wu et al. [2005]. The coupling capacitance between wires $i$ and $j$ is defined as:

$$C_c(i, j) = \alpha \cdot f_{i,j} \cdot \frac{l_{i,j}}{d_{i,j}^{\beta}} \tag{1}$$

where $\alpha$ and $\beta$ are technology-dependent constants, $l_{i,j}$ is the coupling length, $d_{i,j}$ is the wire spacing between nets $i$ and $j$ and $f_{i,j}$ is the switching factor for nets $i$ and $j$. $C_c(i, j)$ is observed to drop quickly as two adjacent nets drift apart. The crosstalk model is thus simplified by assuming that two net segments on different layers and in perpendicular directions are not sensitive. Moreover, $\alpha$ and $f_{i,j}$ are ignored and $\beta$ is assumed to be 2, so the coupling capacitance between two adjacent nets can be estimated simply from $l_{i,j}/d_{i,j}^2$. The following discussion assumes that the space between any two adjacent IRoutes is fixed.

### 2.3. Overview of the Proposed Routing System

This study presents a crosstalk-driven gridless routing system comprising a congestion-driven global router, a crosstalk-driven GTA and an improved NEMO with fast PMT extraction. Crosstalk-driven GTA is undertaken following global routing, and detailed routing is performed in the last stage to connect all disconnected nets. Figure 4 illustrates the proposed routing flow, and the process of postrouting verification and coupling-capacitance calculation. The entire crosstalk-driven GTA comprises global-path merging based IRoute extraction, initial assignment and crosstalk reduction. The IRoute extraction identifies IRoutes for every panel based on the global paths generated by the global router. Since 2-D global routing compresses multiple routing layers of the same preferable routing direction into a layer, every IRoute must be assigned to a specific layer to utilize all routing resources. For simplicity, layer assignment is not specifically invoked to determine the routing layer of every IRoute before GTA. The GTA determines IRoute assignment starting on the first available routing layer, and the remaining incomplete IRoutes are placed on the second available routing layer with the same preferable routing direction once the first routing layer has no available

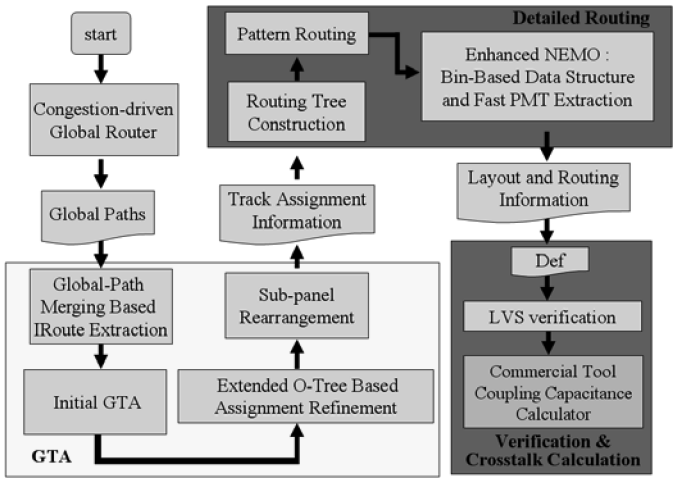



Fig. 4. The proposed crosstalk-driven gridless routing flow and postrouting verification and coupling capacitance calculation.

resource. This process is repeated until the assignment of all IRoutes is complete, or the routing resources are not available. The advantages of this method are simple design, good routing resource utilization and low usage on the top routing layer, thus making more room for next-hierarchy routing.

The proposed GTA algorithm transforms the GTA problem into a constrained floorplanning problem. The following discussion about GTA defines the separation rule between any two adjacent IRoutes as $sp$. All IRoutes are oversized by $sp/2$ to ensure that adjacent IRoutes are always separated. Therefore, the oversized IRoute can neighbor on other IRoutes, and two IRoutes whose projection lines can reach each other are considered to induce coupling capacitance. The GTA problem can be regarded as a special floorplanning problem once the original IRoutes are oversized. An oversized IRoute is equivalent to a block with a fixed $x$-coordinate constraint. The objective of this task is to discover a complete floorplanning of the minimum crosstalk within a fixed-height routing region. The initial IRoute assignment adopts a left-edge-like algorithm to yield an initial assignment quickly. The successive floorplanning-based assignment refinement and subpanel reordering diminish the crosstalk impact of the initial assignment. The extended O-Tree consolidates the available information on the tree by including the overlapping information in the OLG. The total crosstalk is minimized by ripping up every IRoute, and reassigning it into its best position on the extended O-Tree. Subpanel rearrangement is then performed: the current panel is partitioned into several independent subpanels, and the best subpanel sequence is identified to minimize the crosstalk induced on the boundary of every adjacent subpanel.

Detailed routing comprises three steps: routing tree construction, pattern routing and enhanced NEMO with fast PMT extraction. After GTA, most nets are partially routed, and the disconnected components of a net are unconnected pins and preplaced IRoutes. Since detailed routing employs a point-to-point routing scheme, a net with $n$ disconnected components is represented by a routing tree structure organized by $(n-1)$ point-to-point routings. The resulting pattern routing quickly finishes simple routings with line and L-shape patterns. Finally, the enhanced NEMO with bin-based fast PMT extraction connects all incomplete routes. Figure 4 also illustrates the follow-up tasks after detailed routing for verification and crosstalk calculation using commercial EDA tools. The resulting route is first transformed to a DEF-format file and LVS verification

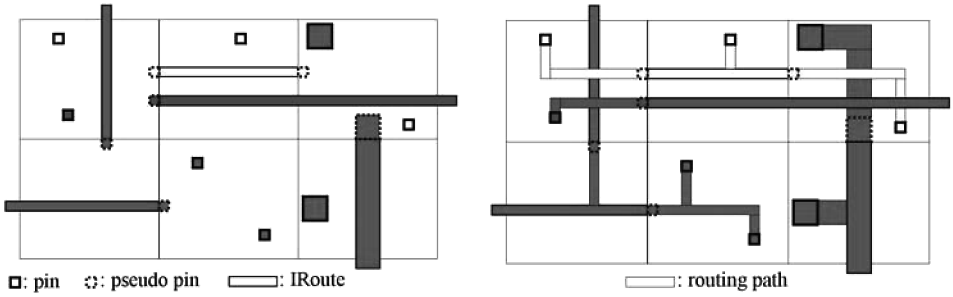



Fig. 5. (a) Partial region (six GCells) of a routing problem; (b) final routing result.

is performed. A placement and routing tool then loads the routing result that passes LVS verification, and computes the total coupling capacitance. Figure 5 shows the partial area (six GCells) of a routing example, where bold and dotted rectangles are pins and pseudo pins, and bold and slim paths are IRoutes and routed paths, respectively. These six GCells contain four nets and eight pins. For connecting pins and IRoutes, six pseudo pins are inserted at both ends of every IRoute. Figure 5(a) only displays the pseudo pins inside these six GCells. Figure 5(a) displays IRoute extraction and assignment results, while Figure 5(b) displays the final routing result, where the paths with bold borders are preplaced IRoutes, and the paths with slim borders connect pins, pseudo-pins and preplaced IRoutes using a detailed router.

## 3. CROSSTALK-DRIVEN GTA

The following discussion considers the assignment on a horizontal routing layer. The crosstalk-driven GTA problem is formulated as follows. Given a fixed outline with a bottom-left corner $(X_L, Y_B)$ and top-right corner $(X_R, Y_T)$, let $\Upsilon = \{ir_1, ir_2, \ldots, ir_n\}$ be a set of $n$ IRotues over-sized by half of the separation rule $(sp/2)$. Each $ir_k$ is represented by the coordinate of its bottom-left-hand corner, width, and height, $(xc_k, y_k, w_k, h_k)$, where the first, third, and last terms are fixed with $xc_k \geq X_L$ and $(xc_k + w_k) \leq X_R$, and the second term is variable. Let $\Lambda = \{b_1, b_2, \ldots, b_m\}$ be the set of existing barriers. The crosstalk-driven GTA problem is to seek an assignment $A = \{y_i | 1 \leq i \leq n\}$ of $y$-coordinates to the lower borders of rectangular over-sized IRoutes such that

(1) $y_i \geq Y_B$ and $(y_i + h_i) \leq Y_T, \forall 1 \leq i \leq n$,
(2) $ir_i \cap ir_j = \phi, \forall i \neq j$ and $1 \leq i, j \leq n$,
(3) $ir_i \cap b_j = \phi, \forall 1 \leq i \leq n$ and $1 \leq j \leq m$, and
(4) $\sum_{i \neq j, 1 \leq i,j \leq n} C_c(i, j)$ is minimized,

where $ir_i \cap ir_j = \phi$ implies $ir_i$ and $ir_j$ do not overlap.

### 3.1. Global-Path Merging Based IRoute Extraction

Previous works on TA do not address the impact of IRoute extraction on routing. Modern global routing algorithms partition a net routing into multiple two-pin routings. Figure 6(a) displays a 5-pin global routing that comprises four two-pin routes in a routing region with 16 GCells. IRoute extraction follows the two-pin routing tree topology. Figure 6(b) displays the extracted IRoutes, where the dotted lines indicate the incomplete routings after GTA that require detailed routing to complete them. This study proposes IRoute extraction based on global-path merging to improve the efficiency of GTA. Observation results reveal that an IRoute can be lengthened to reinforce the efficiency of GTA by merging two global paths of the same direction ending at the

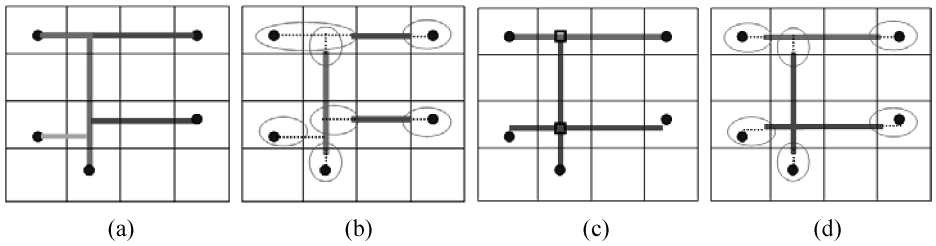



Fig. 6. Two ways for IRoute extraction. (a) and (b) IRoute extraction follows global routing results. Detailed routing requires seven point-to-point routings; (c) and (d) global-path merging yields long IRoutes and less and simple point-to-point routings.

same GCell. Figure 6(c) displays the pink and green IRoutes generated by individually merging two global paths of the same direction ending at the same GCell, highlighted by the bold rectangle. The IRoute extraction result in Figure 6(b) is thus improved by lengthening two horizontal IRoutes, and eliminating one incomplete two-pin routings, as shown in Figure 6(d). In Figure 6(d), five local pin-to-IRoute connections are ignored by track assignment to reserve more routing resources for IRoutes. After all IRoutes are assigned to their tracks, local pin-to-IRoute connections can be connected by pattern routing and maze routing. In a loose GCell, local pin-to-IRoute connections can be completed using straight wires or L-shape patterns. L-shape routing has upper and lower L-shape patterns, and the one with least coupling capacitance is adopted in pattern routing. In a congested GCell, local pin-to-IRoute connections are connected with maze routing. The impact of these short segments ignored in GTA stage is that long segments (IRoutes) have more available routing resources in GTA stage.

### 3.2. Initial GTA

The initial assignment is intended to place as many IRoutes as possible quickly, while minimizing the length of overlap in the maximum clique. The left-edge channel routing algorithm efficiently places as many horizontal nonoverlapping segments in a routing track as possible by sequentially scanning all horizontally segments in order of increasing left-end $x$-coordinates [Sait and Youssef 1999]. The grid-based TA problem is similar to the channel routing problem but without vertical constraints. Accordingly, the left-edge algorithm can be adopted to pack as many nonoverlapping IRoutes into a track as possible for a grid-based TA problem. Since the partial assignment of GTA probably forms uneven borders, the routing region cannot easily be regarded as row by row. Figure 7 presents the proposed initial GTA algorithm. Firstly, the OLGs of all IRoutes are constructed, and then crosstalk minimization in the initial GTA stage is to minimize the effect of the crosstalk in the most congested region: the maximum clique ($cq_{max}$) of the OLG. Crosstalk minimization in the maximum clique firstly assigns the crosstalk effect between any two IRoutes to their connecting edge as an edge cost, and identifies a minimum-weighted Hamiltonian path from $cq_{max}$ (lines 1 and 2). Finally, the order in which every IRoute in $cq_{max}$ is visited along the Hamiltonian path is the order of IRoute assignment in the panel (line 3). The contour of current partial assignment is also constructed using a method proposed elsewhere [Guo et al. 2001] (line 4). For example, the order of the Hamiltonian path for the maximum clique of the OLG in Figure 8 is 10–14–15–17–19–7–16–6. The B-compact assignment for the maximum clique involves placing IRoute 10 at the bottom of the panel and the other IRoutes at the top of previous IRoute following the Hamiltonian path order, as shown in Figure 8. The set of IRoutes with dotted borders is the first maximum clique to be processed.

**Algorithm**: Initial GTA.
**Input**: Set of over-sized IRoutes $\Upsilon$ with left-end $x$-coordinate, width, and height.
**Output**: Initial GTA result.
***begin***
1. Construct the IRoute OLG for this panel and find the maximum clique $cq_{max}$ in OLG;
2. Apply minimum-weighted Hamiltonian path algorithm to determine the IRoute order of $cq_{max}$ with minimum crosstalk effect;
3. Sequentially place all IRoutes in $cq_{max}$ into the panel according to the minimum-weighted Hamiltonian path order to produce a B-compact partial assignment;
4. Maintain current contour;
5. $maxLef = \max(xc_j), \forall ir_j \in cq_{max}\ ;\ \gamma_L = \{ir_i \mid ir_i \notin cq_{max}\ \&\&\ (xc_i + w_i) \le maxLef\}$ ;
6. $minRig = \min(xc_j + w_j), \forall ir_j \in cq_{max}\ ;\ \gamma_R = \{ir_i \mid ir_i \notin cq_{max}\ \&\&\ xc_i \ge minRig\}$ ;
7. Sort IRoutes in $\Upsilon_L / \Upsilon_R$ in decreasing/increasing order of their right/left borders;
8. ***for*** each $ir_i \in \Upsilon_R$
9. check current contour and find the bottommost available position $p_{bm}$ for $ir_i$;
10. ***if*** (there is free space to accommodate $ir_i$) place $ir_i$ to $p_{bm}$ and update current contour;
11. ***for*** each $ir_i \in \Upsilon_L$
12. check current contour and find the bottommost available position $p_{bm}$ for $ir_i$;
13. ***if*** (there is free space to accommodate $ir_i$) place $ir_i$ to $p_{bm}$ and update current contour;
***end***

Fig. 7. Initial GTA algorithm.

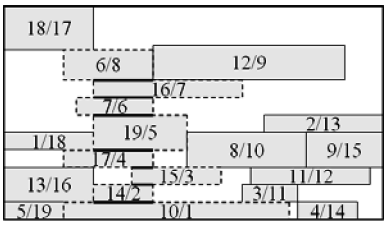



Fig. 8. An initial assignment for 19 IRoutes. Notation $i/r$ signifies that IRoute $i$ is the $r$-*th* IRoute to be placed. The IRoutes with dotted borders form the maximum clique to be first placed.

The bold line between two adjacent IRoutes denotes their overlap length. The number before the slash in each IRoute represents its net number, and that following the slash is referred to as the assignment order.

The remaining unassigned IRoutes are classified into two groups: the IRoutes whose $x$-coordinates of right borders are not larger than the maximum $x$-coordinate of left border of the IRoutes in $cq_{max}$ comprise the left group, $\Upsilon_L$, and the IRoutes whose $x$-coordinates of left borders are not less than the minimum $x$-coordinate of right border of the IRoutes in $cq_{max}$ comprise the right group, $\Upsilon_R$ (lines 5 and 6). IRoutes in $\Upsilon_L$ and $\Upsilon_R$ are sorted in order of decreasing and increasing $x$-coordinates of right borders and left borders, respectively (line 7). For $\Upsilon_R$, in a manner similar to the left-edge algorithm, the unassigned IRoute with the least $x$-coordinate of left border is considered for assignment, and is assigned to its lowest available location. The current contour is also updated (lines 8–10). Similarly, for $\Upsilon_L$, the unassigned IRoutes are also sequentially assigned in order of decreasing $x$-coordinates of their right borders to their lowest available positions (lines 11–13). In Figure 8, $\Upsilon_R = \{12, 8, 3, 11, 2, 4, 9\}$ and $= \Upsilon_L\{13, 18, 1, 5\}$.

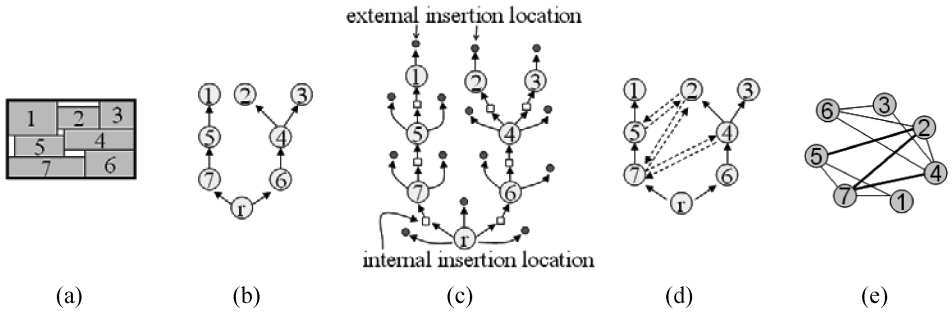



Fig. 9. (a) A B-compact placement; (b) related vertical O-tree; (c) external and internal insertion points of the O-tree; (d) extended vertical O-tree integrated with ordered OLG edges; (e) OLG of the IRoutes in (a).

### 3.3. Extended O-Tree Based Assignment Refinement (EOBAR)

The crosstalk reduction for a horizontal panel forms a restricted floorplanning problem with a fixed $x$-coordinate constraint for every IRoute. The floorplanning representation strongly influences the floorplanning performance. O-Tree [Guo et al. 2001] and B*-tree [Chang et al. 2000] are two well-known methods to represent nonslicing floorplanning. For nonslicing floorplanning, B*-tree is more efficient than O-Tree in terms of runtime and area issues. For GTA, the simulated annealing process is not desired, since many panels are involved in decreasing crosstalk, and the total runtime is unacceptable. GTA is not an admissible placement, as revealed by a GTA result that is a B-compact placement in Figure 9(a). The O-Tree of a floorplan can be constructed as follows. Assume that the placement is a B-compact placement, where no block can be shifted downwards from current position with other blocks fixed. A vertical O-Tree has a root node on the bottom representing the bottom boundary, and a node representing each block. The root node has an edge directed to the nodes, whose bottom borders are located at the bottom boundary. Any two block nodes, say $b_i$ and $b_j$, have an edge from $b_i$ to $b_j$ if $b_j$ abuts $b_i$ on its top.

A fast and deterministic method to reduce crosstalk effect is to eliminate each IRoute iteratively, and insert it in another position with crosstalk reduction profit, as applied to nonslicing floorplanning in Guo et al. [2001]. Figure 9(c) shows external and internal insertion positions of the O-Tree in Figure 9(b). All internal insertion locations and the external insertion locations adjacent to the boundary are candidates for altering GTA. For every IRoute, say $Ir$, all internal and external insertion locations along the O-Tree path containing $Ir$ are evaluated to yield the best insertion point to reduce the crosstalk effect. The EOBAR stage is completed once every IRoute has been evaluated in increasing order of IRoute length and adjusted to its best location. The EOBAR algorithm is displayed in Figure 10, where the extended O-Tree is introduced later. The procedure *Tentative_plow* in the algorithm realizes the evaluation process of IRoute removal and insertion in the O-Tree. The tentative plow removes an IRoute from the O-Tree, places it at a specified location, and reports the new crosstalk effect if the new assignment is legal, that is, if the assignment's height is not greater than the panel's height. To accommodate the inserted IRoute, the tentative plow pushes away the IRoutes that are around the insertion location and overlap the inserted IRoute. The push force may then propagate to the IRoutes on the neighboring O-Tree paths until the emptied space can accommodate the inserted IRoute. The notation adopted in the tentative plow is introduced below.

*Definition Interfering IRoute*. One IRoute is the interfering IRoute of another IRoute, say $Ir$, if it has one edge connecting $Ir$ in OLG, and they are not on the same O-Tree

```
Algorithm: Extended O-Tree Based Assignment Refinement
Input: Initial track assignment result.
Output: New assignment with reduced crosstalk effect.
begin
 1. Construct extended vertical O-tree (EVO_T) for initial assignment;
 2. for each IRoute Ir in increasing order of IRoute length {
 3.     Min_xtalk = xtalk of current assignment;
 4.     for each possible insertion position Ins_pos {
 5.         (New_xtalk, new assignment) = Tentative_plow(EVO_T,Ir,Ins_pos);
 6.         if (New_xtalk < Min_xtalk) {
 7.             Min_xtalk = New_xtalk;  Set new assignment as the best assignment;
 8.         }
 9.     }
10.     if (Min_xtalk < xtalk of current assignment)
11.         replace current assignment with the best assignment;
12. }
end
```

Fig. 10. Algorithm of extended O-tree based assignment refinement.

path. The interfering IRoutes of a plowed IRoute are the potential IRoutes that will interfere with the plowing action.

*Definition IRoute under Test* (*IRT*). The IRoute that is removed from the current location and seeks the insertion location with most crosstalk reduction profit.

*Definition IRT Path*. The O-Tree path containing IRT before removing IRT from O-Tree.

*Definition Seed IRoute for Plowing* (*SIP*). SIPs are classified into three types. A type 1 SIP is on the IRT path and adjacent to the insertion location. A type 2 SIP is around the insertion location, and overlaps the IRT. A type 3 SIP is the first IRoute imposed by the plowing effect propagating from one O-Tree path to another in order to accommodate the inserted IRT. These three types of SIPs and their plow operations are described as follows.

*Type 1*. Type-1 SIP is located on the same O-Tree path as IRT. Two IRoutes between the insertion location of IRT are candidate type-1 SIPs. In the first subgraph of Figure 11(a), IRoutes $A$ and $B$ are type-1 SIPs. IRoute $A$ is first kept unchanged, and IRoute $B$ is plowed downwards. If IRoute $B$ cannot be shifted down by a distance of the height of IRT, then IRoute $A$ must be also plowed upwards to empty out the space reserved for IRT. In the second subgraph of Figure 11(a), IRT is located above its insertion location, which is between two IRoutes $A$ and $B$ (where $B$ is on top of $A$). In his case, only IRoute $B$ is type-1 SIP. IRoute $A$ remains unchanged since the original assignment is B-compact, and IRoute $B$ is plowed upwards by a distance of the height of IRT.

*Type 2*. Type-2 SIP is located on a different O-Tree path from IRT and is close to the location of insertion of IRT, such that type-2 SIP overlaps IRT after IRT is placed at this insertion location. In Figure 11(b), if IRT is moved downwards and inserted between IRoutes $A$ and $B$, then IRoute $B$ is a type-1 SIP and is uplifted. After the upward plow on type-1 SIP, $B$, is completed, IRT is placed at its insertion location. At this time, IRoutes $C$ and $D$ are the interfering IRoutes of IRT (overlap with IRT and are

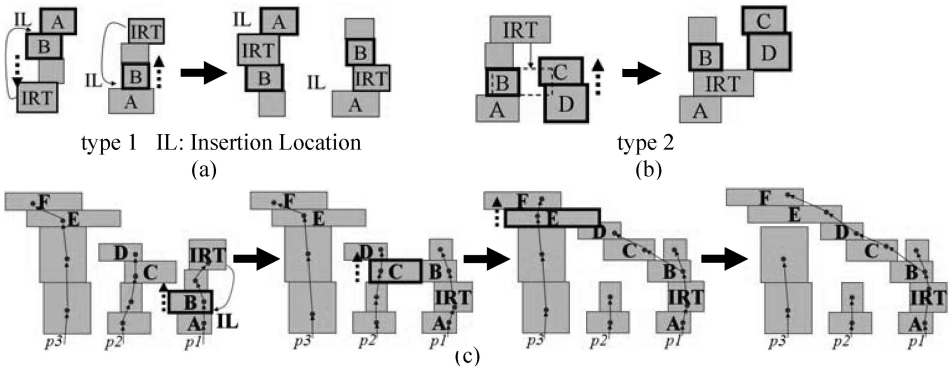



Fig. 11. Three types of plowing. (a) type 1; (b) type 2; (c) type 3.

on the O-Tree path that is different from that of IRT), and are regarded as type-2 SIPs. Since IRoutes *C* and *D* are on the same O-Tree path and IRoute *D* is above IRoute *C*, uplifting IRoute *C* also uplifts IRoute *D*. Thus IRoute *C* is uplifted. Finally, IRoute D is uplifted to abut on the top side of IRT, as shown in the right subgraph of Figure 11(b).

*Type 3*. When an uplifted SIP is blocked by another IRoute that is located on another O-Tree path, a type-3 SIP is defined. In Figure 11(c), IRoute *C* hinders the uplifting of IRoute *B*, the IRoute (IRoute *C*) that hinders an upward plow operation must also be plowed upwards. In Figure 11(c), IRoute *B* is on O-Tree path *p1*, while IRoute *C* is on another O-Tree path *p2* and is a type-3 SIP. Notably, the plow operation for a type-3 SIP probably triggers another type-3 SIP plow operation. In the third subgraph of Figure 11(c), the plow operation for type-3 SIP *C* is obstructed by IRoute *E*, which is on another O-Tree path *p3*. To complete the plow operation of type-3 SIP *C*, a new upward plow operation to raise the new type-3 SIP *E* is initiated. The last subgraph of Figure 11(c) displays the final plowing result. Following the plow operation, the O-Tree is reconfigured as follows. IRoute *B* adds a new edge that points to IRoute *C*, and IRoute *D* adds a new edge that points to IRoute *E*. The original incident edges of IRoutes *C* and *E* are eliminated: IRoutes *C* and *E* finally abut the tops of IRoutes *B* and *D*, respectively. The plow operation for type-3 SIP proceeds necessarily upwards since the original assignment is B-compact and downward plowing on a B-compact assignment is infeasible.

Downward plow is only applied to the type 1 SIP, and the type 2 and 3 SIPs are invoked only by upward plow. Figure 12 shows the tentative plow algorithm. Tentative plow first identifies the type 1 SIPs, and then completes plow operation on type 1 SIPs. If the plowing on type 1 SIPs is blocked, then type 3 SIP is identified and the plowing on type 3 SIP is initiated. After the plowing on type 3 SIP is completed, type 1 SIPs can then be uplifted to their desired positions. At this time, IRT can be placed at its new location. The type 2 SIP is then identified, and new plow is invoked on the type 2 SIP if IRT overlaps any IRoute on neighboring O-Tree path. If the type 2 SIP requires upward plowing, then the existence of interfering IRoute (type 3 SIP) must also be checked, and an identified type 3 SIP initiates a new upward plowing on the neighboring O-Tree path.

For a downward plow, the final location of a falling IRoute is determined by its parent IRoute and the interfering IRoute below and closest to it. If its parent IRoute's location is higher than that of interfering IRoute, then the IRoute remains at the top side of its parent IRoute; otherwise, the IRoute moves to the top side of interfering IRoute. Since a falling IRoute cannot determine its location until the location of its parent IRoute is fixed, a recursive operation is conducted to finish the downward plow. For an upward

```
Algorithm: Tentative_plow
Input: extended O-tree (EOT), inserted IRoute (IRT), insertion location.
Output: the new assignment and its crosstalk effect.
begin
1.  Identify and store the type-1 SIP in SIP_List;
2.  while (SIP_List is not empty) {
3.    SIP = SIP_List.next( );      D_plow = GetPlowDist(SIP);
4.    if (SIP is type 1) {
5.      if (IRT is above its insertion location)
6.        plow SIP upwards with a distance of D_plow and do upward plow on type 3 SIP if
      necessary;
7.      else { // IRT is below its insertion location
8.        plow SIP downwards as far as possible;
9.        if (plowing distance of SIP < D_plow ) {
10.         SIP = SIP's child in EOT;
11.         plow SIP upwards with a distance of (D_plow – plowing distance of prior SIP)
      and do upward plow on type 3 SIP if necessary;
12.       }
13.     }
14.     Identify and store type-2 SIP in SIP_List;
15.   }
16.   else // SIP is type 2
17.     plow SIP upwards to abut the newly placed IRT at its top side, and do upward plow
      on type 3 SIP if necessary;
18. }  // end of while
19. if (new assignment's height > panel height)
20.   return MAX_INT and no feasible assignment;
21. else {
22.   calculate the crosstalk (Xtalk_new) of new assignment;
23.   return Xtalk_new and new assignment;
24. }
end
```

Fig. 12. Algorithm of IRoute tentative plow.

plow, the final location of an uplifted IRoute is derived by the desired plowing distance and its parent IRoute's location. Thus a breadth-first search is favorable to upward plow.

An efficient method for identifying an IRoute's interfering IRoutes is essential to maximizing the performance of a tentative plow. These IRoutes can be identified and access through OLG. However, $m$ insertion points and an OLG of $n$ vertices require access to $m \times (n-3)$ vertices since for every IRT, the IRT, its parent, and its child do not need to be checked. To lower the overhead of accessing OLG, the O-Tree is integrated with ordered OLG, which is defined as follows. The edges of a vertex are stored in increasing order of IRoute's $y$-coordinate. An OLG edge is regarded as *essential* if two associated IRoutes of its two connected nodes are located on different O-Tree paths. An OLG edge that connects to the IRoutes on the same O-Tree path is disregarded since the OLG edges are adopted to investigate the interfering IRoutes that will not appear on the same O-Tree path. Figure 9(d) shows the extended O-Tree with ordered OLG, where the original OLG is shown in Figure 9(e). The dotted lines in Figure 9(d) denote the essential OLG edges. In Figure 9(e), IRoute 7 connects to four IRoutes using four OLG edges, two of which (IRoutes 1 and 5) are on the same O-Tree path as IRoute 7,

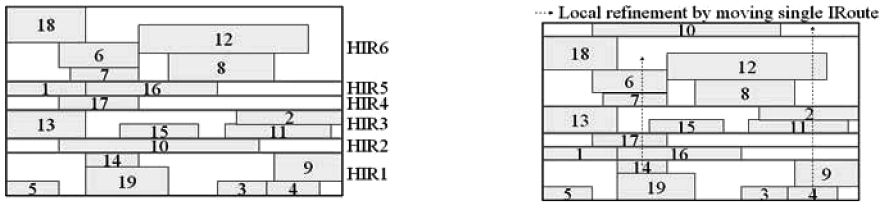



Fig. 13. (a) Assignment, produced by EOBAR, is partitioned into six subpanels with cut lines; (b) The assignment result following subpanel reordering. The dashed arrow line implies that the assignment can be further upgraded through local refinement (move single IRoute).

and two of which (IRoutes 2 and 4) are on a different O-Tree path. The edges in bold in Figure 9(e) are essential edges. IRoute 7 has two OLG edges with the order of (7, 4) and (7, 2). Similarly, IRoute 2 has two OLG edges in the order of (2, 7) and (2, 5). The removal of IRoute 1 and its insertion into the bottom of IRoute 7 is illustrated as follows. IRoute 7 is a type 1 SIP and is uplifted. To find its interfering IRoute, its two ordered OLG edges are investigated and edge (7, 4) is checked first to identify IRoute 4 as its interfering IRoute and a type 3 SIP. Accordingly, IRoute 4 is also uplifted and finally abuts the top of IRoute 7. Another example is the removal of IRoute 7 and its insertion between IRoutes 1 and 5. IRoute 5 is identified as a type 1 SIP, and plowed downwards. In this case, IRoute 5 is successfully moved to the bottom of the current panel. IRoute 7 is then placed at the desired position. However, IRoute 7 overlaps an IRoute on the adjacent O-Tree path, and IRoute 4 is identified as a type 2 SIP. IRoute 4 is then uplifted to solve the overlap with IRoute 7.

### 3.4. Subpanel Rearrangement

In the EOBAR stage, crosstalk reduction only considers the effect between individual IRoutes. Further reduction can be attained by considering the effects between subpanels. Figure 13(a) shows an assignment after performing EOBAR, which is partitioned into six subpanels with five cut lines. Every subpanel is regarded as a hyper-IRoute (HIR), probably with two different contours. For instance, in Figure 13(a), HIR1, HIR3, and HIR6 have different contours on their top and bottom sides, but the others do not. Reordering of these HIRs yields different overlap lengths. The HIR OLG is defined like the IRoute OLG, where every HIR is represented as a node in the graph. There are two directed edges between any two nodes, since HIR is not considered in a mirrored state in this work. A directed edge from a node, say $A$, to another node, say $B$, implies that HIR $A$ is placed on the bottom of HIR $B$. Its edge cost is the overlap length of the assignment sequence A$\rightarrow$B (where the symbol $\rightarrow$ implies "on bottom of"). The crosstalk minimization problem is then transformed into the Minimum Weighted Hamiltonian Path (MWHP) problem. Finding an MWHP on the clique of IRoute OLG commences with the longest IRoute as the initial node, and selects one adjacent unvisited node with the lowest edge cost out of all the adjacent unvisited nodes. The newly selected node serves as a new initial node, and proceeds similarly until all nodes are visited. This work employs the similar MWHP algorithm, and elevates using the branch and bound algorithm.

The branch-and-bound search tree is a multi-ary tree. Each generated node enrolls the total overlap length of the incomplete Hamiltonian path from the root to the present node as the overlap-length lower bound of all paths with the same incomplete partial sequence identified thus far. The sums of all inward edges of all nodes are first computed and sorted in decreasing order to determine the root order of the search trees. The gain

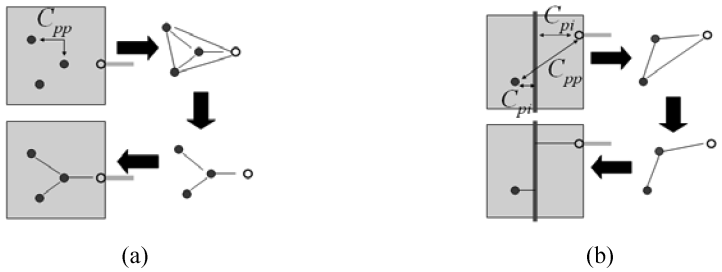



Fig. 14. Two edge costs for seeking minimum spanning tree: (a) using the Manhattan distance between any two pins/pseudo pins; (b) using the cost of pin/pseudo pin to IRoute based on their projection distance.

of this ordering is avoiding the largest overlap length in the first node probably gets a good assignment, which is a good upper bound for fast dropping worse assignment in successive search. Each node visits its unvisited adjacent node of the least edge cost as the search process proceeds. As the depth-first search continues, the first identified assignment has good quality similar to that in Ho et al. [2005], and can behave as a good upper bound for fast convergence. Experimental results indicated that this algorithm can search the optimal solution within 0.001 seconds for most test cases.

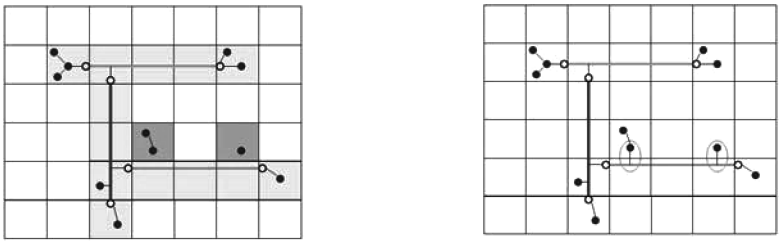
Figure 13(b) displays the subpanel reordering result of Figure 13(a). In this case, the branch-and-bound algorithm can identify the optimal solution with overlap length 21, as compared to the overlap length of 28 obtained using heuristics MWHP algorithm in [Ho et al. 2005]. Moreover, if the original panel is loose, then an individual IRoute can be moved to decrease the overlap length. For instance, in Figure 13(b), IRoutes 4 and 14 can be moved up, while IRoute 11 can be moved down.

The proposed crosstalk-driven GTA minimizes the total crosstalk. Crosstalk minimization on critical nets is sometimes required. The proposed crosstalk-driven GTA can thus be extended as follows. The IRoutes of a critical net are called *critical IRoutes*. In EOBAR stage, the critical IRoutes in a panel are first removed and then inserted at possible insertion positions to find their minimum-crosstalk assignment in order of declining criticality. Adjacency refers to the relation between two neighboring IRoutes. Removing an IRoute yields a new adjacency that is defined in terms of the top and bottom neighboring IRoutes of the removed IRoute. Furthermore, inserting an IRoute between two IRoutes, say *IRa* and *IRb*, yields two adjacencies that are the new neighboring relationships between *IRa* and the inserted IRoute, and between the inserted IRoute and *IRb*. Every processed critical IRoute is inserted between two IRoutes that form the new adjacency to determine whether its crosstalk effect can be further reduced. Accordingly, every critical IRoute can try every new adjacency to minimize the crosstalk of individual net.

## 4. DETAILED ROUTING

### 4.1. Routing Tree Construction

Each net routing following GTA leaves several IRoutes and unconnected pins. Planning how to connect pins and IRoutes together is stipulated to achieve good routing quality. Generally, several IRoutes are derived based on the global path. Two pseudo pins are defined for each IRoute in the two ending GCells of a straight global path. The remaining work for detailed net routing is to connect all pins, pseudo pins and IRoutes together. Figure 14 shows two cases of routing tree construction. In Figure 14(a), all pins and pseudo pins are regarded as nodes, and any two nodes have an edge with a cost defined by their Manhattan distance. This cost, called the Manhattan distance cost, of any two pins or pseudo pins, is denoted as $C_{pp}$. The other edge cost defines the edge



(a) (b)

Fig. 15. Two steps of routing tree construction: (a) This routing tree is derived after identifying the minimum spanning trees in all GCells with one IRoute passing by; (b) final routing tree is derived after seeking the minimum spanning tree of the routing tree in (a) and all GCells without IRoute passing by.

cost of pin or pseudo pin to IRoute. Figure 14(b) displays a case of a GCell containing a pin, a pseudo pin and an IRoute. An IRoute is also denoted as a node in the complete graph, but its edge cost, represented as $C_{pi}$, between itself and any pin or pseudo pin is shortest distance between them. For instance, the edge cost between the IRoute and the pin or pseudo pin in Figure 14(b) is given by the horizontal distance from the pin or pseudo-pin to the projection point on the IRoute. Prim's algorithm is adopted to seek the minimum spanning tree. This routing tree determines the point-to-point routings invoked by detailed routing.

The routing tree found at this point probably does not include all pins of a net, since some segments are too short to be processed by GTA. For instance, there are two short segments not processed by GTA, as shown with two heavy shadow GCells in Figure 15(a), and its routing tree is not complete. To complete the entire routing tree construction, the routing tree identified so far and all pins in a short-segment GCell are treated as nodes, and the edge cost of a pin to the existing routing tree is the shortest distance between them. For each short-segment GCell, a complete graph is constructed and minimum spanning tree is discovered. The final routing tree is then completed, as shown in Figure 15(b).

### 4.2. Pattern Routing

Pattern routing has already been adopted in global routing to increase the routing speed [Kastner et al. 2000; Pan and Chu 2006]. This work conducts pattern routing before detailed routing to simplify some easy routings. Detailed routing can solve all point-to-point routings after building the routing tree. However, the point-to-point routings are totally different from those in a two-stage routing flow, that is, global routing plus detailed routing. Long-distance routing has been completed by GTA, and the detailed router is responsible for short-distance routing and incomplete IRoutes. Observations on some real circuits, many short-distance routings are based on pin-to-IRoute routing. Most such routings can be completed using a direct or L-shape connection. Therefore, simple direct connection and L-shape routing is undertaken at this stage.

### 4.3. Bin-Based Data Structure and Fast PMT Extraction

The detailed routing is dealt with by NEMO in this work. NEMO performs gridline reduction to construct a simplified routing graph. Gridline reduction tends to impose a searching penalty as the number of obstacles inducing a gridline rises. The bin-based data structure has been widely adopted in physical design. If the bin size is set equal to the GCell size, then the bin-based storing scheme simplifies gridline generation, since gridlines are naturally reduced by simply extending the borders of obstacles in the GCells (bins) of the global path. Figure 16 shows a routing region comprising

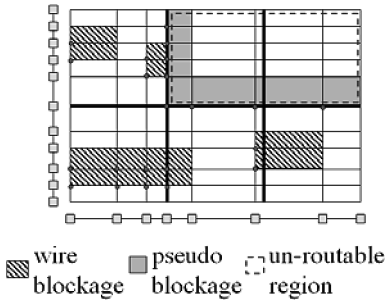



Fig. 16. This routing region contains $3 \times 2$ GCells (bins). The global path is the L-shape, and the right two GCells on top row are un-routable. All tiles within obstacles are marked as blocked by placing a circle on the left bottom corner in the figure. PMT extraction only requires checking the states of the queried tile and its left/bottom and right/top successive tiles for a horizontal/vertical layer graph.

Table I. Benchmark Circuits Statistics for Full-Chip Routing

| Circuit | Size ($\mu m$) | # 2-pin nets | # Pin | # GC | # panel |
|---|---|---|---|---|---|
| S5378 | $4350 \times 2390$ | 3124 | 4818 | $55 \times 30$ | 85 |
| S9234 | $4040 \times 2250$ | 2774 | 4260 | $51 \times 28$ | 79 |
| S13207 | $6600 \times 3650$ | 6995 | 10776 | $83 \times 46$ | 129 |
| S15850 | $7050 \times 3890$ | 8321 | 12793 | $89 \times 49$ | 138 |
| S38417 | $11440 \times 6190$ | 21035 | 32344 | $144 \times 78$ | 222 |
| S38584 | $12950 \times 6720$ | 28177 | 42931 | $163 \times 85$ | 248 |

$3 \times 2$ GCells (bins). The bold lines define GCells and bins. The global path is an L-shape, and the rightmost two GCells on the top row are not allowed for this routing. To speed up PMT extraction, pseudo-blockages are inserted around the global path to restrict identified PMTs within the global path. Moreover, NEMO dynamically identifies PMTs during path propagation. Although the slit and interval tree is an efficient data structure, it inevitably requires numerous queries before reaching the target. Hence, if PMTs can be identified without visiting the slit and interval tree, then path propagation can be performed more efficiently than previously. Since the point array has been established, as indicated by the series of rectangles on the bottom and left side of the routing region, all bins are scanned sequentially to set as *blocked* the tiles covered by obstacles within each bin. All tiles outside and neighboring the global path are also set as blocked. In Figure 16, the tiles whose left bottom corners are marked with a circle are set blocked. Identifying PMTs becomes simple and straightforward once all tiles within obstacles are set blocked and the others are set *unvisited*. For a horizontal layer graph, if a queried tile is in the unvisited state, then PMT extraction groups its left and right successive unvisited tiles until a routing boundary or blocked tile are reached. This operation is fairly simple and fast, and the newly proposed PMT extraction makes path propagation performed more efficiently than previously.

## 5. EXPERIMENTAL RESULTS

All routing tests were conducted on a 1.2GHz Sun Blade-2000 workstation with 2GB memory with six MCNC benchmark circuits as presented in Table I. All cases were routed using three routing layers. In the table, "# 2-pin nets" denotes the number of two-pin connections after net decomposition; "# GC" denotes the number of rows and columns after the chip is partitioned into global cells in the global routing stage, and "# panel" represents the number of panels processed during GTA stage. We first compare the routing results of the proposed enhanced NEMO detailed router and NEMO detailed router to show the efficiency of the proposed method. Then, we compare

Table II. Comparison of Routing Performance between NEMO and This Work

| | NEMO | | This work | |
|---|---|---|---|---|
| | Time (Tn:sec) | Mem (MB) | Time (Tt:sec) | Mem (MB) |
| s5378 | 2.4 | 10 | 2.10 | 22 |
| s9234 | 1.7 | 9 | 1.45 | 21 |
| s13207 | 6.6 | 15 | 4.43 | 25 |
| s15850 | 8.8 | 18 | 6.51 | 27 |
| s38417 | 37.2 | 48 | 13.46 | 37 |
| s38584 | 73.7 | 66 | 30.52 | 45 |

Table III. Routing Runtime Statistics of This Work

| | Proposed three-stage gridless routing system (TSGRS) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Ini. (1) | | GTA C.R. (2) | | Preprocess (3) | | Pattern routing (4) | | (1) + (2) + (3) + (4) | | Enhanced NEMO | |
| | **FR** | VR | **FR** | VR | **FR** | VR | **FR** | VR | **FR** | VR | **FR** | VR |
| s5378 | **0.05** | 0.04 | **0.77** | 0.92 | **0.15** | 0.12 | **0.08** | 0.06 | **1.05** | 1.14 | **1.07** | 1.91 |
| s9234 | **0.04** | 0.02 | **0.45** | 0.44 | **0.13** | 0.10 | **0.06** | 0.06 | **0.68** | 0.62 | **0.86** | 1.12 |
| s13207 | **0.12** | 0.09 | **0.49** | 0.26 | **0.29** | 0.21 | **0.21** | 0.18 | **1.11** | 0.74 | **1.19** | 3.66 |
| s15850 | **0.16** | 0.11 | **2.55** | 2.97 | **0.40** | 0.28 | **0.31** | 0.27 | **3.42** | 3.63 | **2.07** | 4.02 |
| s38417 | **0.42** | 0.28 | **7.33** | 6.40 | **0.99** | 0.65 | **0.93** | 0.71 | **9.67** | 8.04 | **5.86** | 9.82 |
| s38584 | **0.59** | 0.39 | **11.83** | 11.66 | **1.82** | 1.05 | **1.67** | 1.27 | **15.91** | 14.37 | **8.73** | 15.85 |

**Ini.**: initialization for GTA; **GTA C.R.**: three-phase GTA crosstalk reduction; **Preprocess**: preprocessing for detailed routing.

the routing results of the proposed three-stage routing system and a two-stage routing system (global routing + NEMO detailed router) in Li et al. [2007] to display the improved performance by the proposed three-stage routing system. Finally, we compare the crosstalk minimization statistics of the proposed three-stage routing system and a commercial Place and Route tool to show the efficiency of the designed crosstalk minimization flow.

### 5.1. Enhanced Implicit Connection-Graph-Based Detailed Routing

To determine the efficiency of the proposed method for improving NEMO, all cases were routed by global and detailed routers. Detailed routing was completed with NEMO and the proposed enhanced NEMO. Table II displays the detailed-routing performance comparison between NEMO and this work. The memory required by bin-based data structure increased slowly to gain the superiority over slit and interval tree for the last two circuits. Meanwhile, the enhanced router had 1.72 times the runtime speedup of NEMO on average. Since the proposed method only adjusts the methods of gridline extraction and PMT extraction, all routing cases are finished with the same routing quality as NEMO.

### 5.2. Gridless Routing System

The entire routing system comprises a global router, crosstalk-driven GTA, and enhanced NEMO. Table III lists the routing runtime statistics of this work for fixed- and variable-rule routings. Variable-rule routing was performed using the same circuits with modified rule set as follows. The width of the longest 10% of nets was doubled; that of the next 10% of nets was multiplied by 1.5, while the others remain unaltered. Since many pins on the first metal layer are aligned and set apart in minimum-rule space, and design rule violation occurs if the rule of the first metal layer and its pins are widened, the rules of the first metal layer remained unchanged. The runtime is split into five stages, namely initialization for GTA, GTA, preprocessing before detailed routing, pattern routing, and detailed routing. Table IV compares the routing statistics of this work and that of Li. et al. [2007] for fixed- and variable-rule routings.

Table IV. Comparison of Routing Results between This Work (three-stage routing system) and Li et al. [2007] (two-stage routing system - G.R. + NEMO)

| | Total run time (sec) | | | | W.L. ($\mu$m) | | | |
|---|---|---|---|---|---|---|---|---|
| | This work | | Li et al. [2007] | | This work | | Li et al. [2007] | |
| | **FR** | VR | **FR** | VR | **FR** | VR | **FR** | VR |
| s5378 | **2.12** | 3.05 | **2.4** | 3.74 | **8.1e4** | 8.2e4 | **7.4e4** | 7.6e4 |
| s9234 | **1.54** | 1.74 | **1.7** | 2.69 | **6.0e4** | 6.1e4 | **5.5e4** | 5.6e4 |
| s13207 | **2.30** | 4.40 | **6.6** | 11.28 | **1.9e5** | 1.9e5 | **1.7e5** | 1.8e5 |
| s15850 | **5.49** | 7.65 | **8.8** | 16.33 | **2.4e5** | 2.4e5 | **2.2e5** | 2.2e5 |
| s38417 | **15.53** | 17.86 | **37.2** | 56.36 | **5.2e5** | 5.2e5 | **4.8e5** | 4.9e5 |
| S38584 | **24.64** | 30.22 | **73.7** | 143.39 | **7.2e5** | 7.3e5 | **6.7e5** | 6.8e5 |
| Comp. | **1** | 1 | **2.02** | 2.56 | **1** | 1 | **0.92** | 0.93 |

Table V. Statistics of Completed Iroutes and Point-to-Point Routings by GTA and Pattern Routing

| | # Iroutes / # incomplete Iroute | | # completed by pattern routing / # point-to-point routings (completion rate) | |
|---|---|---|---|---|
| | **FR** | VR | **FR** | VR |
| s5378 | **1705/0** | 811/5 | **3117/4823 (64.6%)** | 1839/3915 (47.0%) |
| s9234 | **1309/0** | 575/1 | **2637/4076 (64.7%)** | 1550/3330 (46.5%) |
| s13207 | **3490/0** | 1698/13 | **6446/10469 (61.6%)** | 3959/8656 (45.7%) |
| s15850 | **4180/19** | 1999/37 | **8073/12449 (64.8%)** | 4692/10203 (46.0%) |
| s38417 | **9779/9** | 4440/13 | **19661/30745 (63.9%)** | 11825/25323 (46.7%) |
| s38584 | **13219/12** | 5943/84 | **25946/41263 (62.9%)** | 15302/37747 (45.3%) |

The work of Li et al. [2007] is a two-stage routing system (global routing + detailed routing). The runtime speedup for six test cases is in the range 1.1–2.99 times and 1.23–4.74 times for fixed- and variable-rule routings, respectively. This work gains a runtime speedup of 2.02 and 2.56 times on average for fixed- and variable-rule routings. The runtime speedup is mainly influenced by GTA. The GTA runtime in three large designs (s15850, s38417, and s38584) dominates the whole routing runtime. The wire length is getting longer by 7% and 8% for crosstalk minimization for fixed- and variable-rule routings. All cases were 100% completed under two rule sets. Table V lists the statistics of completed IRoutes and point-to-point routings by GTA and pattern routing. GTA completes the placement of all IRoutes of three out of six test cases for fixed-rule routing. For variable-rule routing, the incomplete IRoute number is in the range 1–84 and total 153 IRoutes were incomplete by GTA, resulting in more runtime by detailed routing in these cases. For fixed- and variable-rule detailed routings, about 64% and 46% point-to-point routings were completed with pattern routing to lower the burden of detailed routing. High completion rate before detailed routing was realized through IRoute assignment, followed by connecting vicinal pins.

### 5.3. Crosstalk Minimization

Crosstalk minimization in this work is achieved by crosstalk-driven GTA. Table VI shows the crosstalk reduction, for fixed- and variable-rule routings in the GTA stage. GTA comprises three stages, initial assignment, O-Tree based iterative reassignment and HIR reordering. In Table VI, columns 2, 4, and 6 compare the coupling capacitance of initial assignment, O-Tree based refinement and HIR reordering results for fixed-rule routing, and columns 3, 5, and 7 list the coupling capacitance of initial assignment, O-Tree based refinement and HIR reordering results for variable-rule routing. For fixed-rule routing, GTA contributes 57–63% coupling capacitance reduction (60% on average), while a coupling capacitance reduction of 36–56% is achieved by GTA for variable-rule routing (44% on average). The second experiment in crosstalk minimization addresses the variation in crosstalk for different IRoute orderings in EOBAR stage for fixed-rule routing. IRoutes are processed in order of decreasing IRoute length. Three

Table VI. Statistics of Crosstalk Reduction for Fixed- and Variable-Rule Routings

| Circuit | Initial assignment Coupling cap. × $10^3$ (C1) | | O-tree based refinement Coupling cap. × $10^3$ (C2) | | HIR rearrangement Coupling cap. × $10^3$ (C3) | | Total RR (%) | |
|---|---|---|---|---|---|---|---|---|
| | FR | VR | FR | VR | FR | VR | FR | VR |
| S5378 | **.168** | .123 | **0.108** | 0.074 | **.069** | .066 | **59** | 46 |
| S9234 | **.107** | .086 | **0.072** | 0.049 | **.040** | .038 | **63** | 56 |
| S13207 | **.379** | .294 | **0.374** | 0.266 | **.160** | .186 | **58** | 37 |
| S15850 | **.493** | .363 | **0.344** | 0.260 | **.210** | .231 | **57** | 36 |
| S38417 | **1.013** | .794 | **0.626** | 0.453 | **.372** | .394 | **63** | 50 |
| S38584 | **1.402** | 1.026 | **0.896** | 0.675 | **.557** | .607 | **60** | 41 |
| Ave. | | | | | | | **60** | 44 |

·Coupling cap.: $l_{i,j}/d_{i,j}^2$; FR: fixed rule; VR: variable rule;

·RR: reduction rate = (C1-C3)/C1;

Table VII. Comparison of Fixed-Rule Detailed Routing Results of a Commercial Routing Tool and This Work

| Circuit | Run time (sec) | | Wire length (WL) ($\times 10^4 \mu$m) | | Coupling capacit. (pf) | |
|---|---|---|---|---|---|---|
| | This work | CT wt SI | This work | CT wt SI/CT wt no SI | This work | CRT wt SI |
| s5378 | 2.12 | 13 | 8.1 | 7.7/7.7 | 3.76 | 4.67 |
| s9234 | 1.54 | 11 | 6.0 | 5.7/5.7 | 2.36 | 2.72 |
| s13207 | 2.30 | 43 | 19 | 18/18 | 8.38 | 9.74 |
| s15850 | 5.49 | 36 | 24 | 23/22 | 11.5 | 13.77 |
| s38417 | 15.53 | 81 | 52 | 50/49 | 21.86 | 23.68 |
| s38584 | 24.64 | 128 | 72 | 69 /68 | 32.62 | 37.06 |
| Comp. | 1 | 8.16 | 1 | 0.95/0.94 | 1 | 1.16 |

·CT wt SI: commercial routing tool with signal integrity capability.

out of six benchmarks reveal slightly more crosstalk after EOBAR, while the others reveal slightly less crosstalk. On average, the short-IRoute-first policy is associated with 0.5% less crosstalk than the long-IRoute-first policy after EOBAR. Subpanel rearrangement increases the crosstalk of all benchmarks except one when EOBAR is performed in decreasing order of IRoute length. On average, the crosstalk effect following subpanel rearrangement in order of decreasing IRoute length in EOBAR is 1.3% larger than that obtained by rearrangement in order of increasing IRoute length. The experiment demonstrates that the order of IRoute length in EOBAR stage only slightly influences the crosstalk effect after EOBAR.

Since initial assignment does not totally focus on crosstalk reduction, and may cause bad-quality assignment in terms of coupling capacitance, six test cases are also routed with a commercial routing tool enabling signal integrity (SI) capability to compare the efficiency of reducing coupling capacitance. Table VII compares the detailed routing results and coupling capacitance of this work and that of the commercial routing tool for fixed-rule routing. The runtime speedup for six test cases is in the range 5.19–18.7 times (8.16 times speedup on average), while the total wire length has about 5% increasing rate. As regarding the comparison of coupling capacitance, the routing results of this work are translated to DEF files and then fed into the commercial routing tool. The built-in coupling capacitance estimator of the commercial routing tool is employed to calculate the coupling capacitance of this work and that of the commercial routing tool. This work yields less coupling capacitance than the routing tool in all cases with the reduction rate in the range 7.7–19.5% ((1.16–1)/1.16 = 13.8% on average). The commercial routing tool split a long net into several wire segments routed in different tracks, however the proposed track assignment algorithm does not consider wire breaking currently. Wire breaking probably provides better connections of

pins to IRoutes, which results in less wire length. Besides, wire breaking can contribute to the further reduction of coupling capacitance.

Wirelength and coupling capacitance are two factors that affect timing. Increasing wirelength may worsen timing while reducing coupling capacitance may improve timing. We attempt to verify the impact of this work on the timings of used benchmarks. The ISCAS benchmarks used in this work are released by Prof. J. Cong, and simplified for the use of academic researches. These benchmarks only contain pseudo cell types, cell positions, pin types, pin positions, and net information. In these benchmarks, each pin is reduced to a point. No cell library is provided for further timing analysis, thus we can not analyze timing for these benchmarks. The IWLS 2005 benchmarks[1] contain the netlists of ISCAS benchmarks as well as a cell library released by Cadence. We applied the used commercial tool to place each ISCAS benchmark, and then generated a new set of benchmarks by extracting the pin positions and net information from the placed design. However, the shape of a pin in IWLS 2005 is a complex polygon, which requires a router to have the capability to perform polygon-to-polygon routing. The commercial tool performs this type of routing very well, but NEMO does not have this capability. Thus NEMO can not complete the routings of the new benchmarks, and the timing analysis is not available, neither. The timing analysis on the results routed by commercial tool shows that WNS is averagely improved by 0.23% even though SI operation enlarges total wirelength by 1.3%. Since this work yields less coupling effect reduction with only 5% longer total wirelength than the used commercial routing tool, we can infer that the timings of benchmarks routed by this work also change slightly.

## 6. CONCLUSIONS

This work presents the first three-stage crosstalk-driven gridless routing system, comprising a congestion-driven global router, a crosstalk-driven GTA and an enhanced implicit connection-graph-based detailed router. Firstly, previous works on grid-based TA do not discuss the impact of IRoute extraction on routing. This study proposes global-path merging based IRoute extraction to decrease the number of point-to-point routings by merging two global paths of the same direction ending at the same GCell. After initial assignment, crosstalk reduction in GTA is transformed to a nonslicing floorplanning problem, and an O-Tree-based deterministic floorplanning algorithm is employed. Further reduction is obtained by splitting the routing region into HIRs, then reordering the HIRs with a branch and bound algorithm. After GTA and routing tree construction, many original point-to-point routings are set to connect to IRoutes, and can be simply resolved using pattern routing. Finally, the detailed router simplifies its gridline extraction and PMT extraction by adopting a bin-based data structure and tagging blocked tiles. Experimental results reveal that the enhanced bin-based implicit connection-graph based router has 1.72 times the runtime speedup of NEMO on average. Besides, the proposed three-stage crosstalk-driven gridless routing system can perform over 2.02 times faster for fixed- and variable-rule routings than an implicit connection-graph-based router, NEMO. As compared with a commercial routing tool enabling signal integrity optimization, this work yields an average runtime speedup of 8.16 times and an average 13.8% reduction rate in coupling capacitance calculated with the built-in coupling capacitance estimator of the commercial routing tool, with 5% wire-length increasing rate. For further crosstalk and wire-length reduction, wire breaking has to be considered in track assignment to develop the flexibilities of lowering overlap length of adjacent wires and of connecting pins to IRoutes.

[1](http://www.iwls.org/iwls2005/benchmarks.html)

## REFERENCES


BATTERYWALA, S., SHENOY, N., NICHOLLS, W., AND ZHOU, H. 2002. Track assignment : A desirable intermediate step between global routing and detailed routing. In *Proceedings of the IEEE International Conference on Computer Aided Design*. 59–66.

CHANG, C. C. AND CONG, J. 1999. An efficient approach to multilayer layer assignment with an approach to via minimization. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 18*, 5, 608–620.

CHANG, C. C. AND CONG, J. 2001. Pseudopin assignment with crosstalk noise control. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 20*, 5, 598–661.

CHANG, Y. C., CHANG, Y. W., WU, G. M., AND WU, S. W. 2000. B*-trees: A new representation for non-slicing floorplans. In *Proceedings of the ACM/IEEE Design Automation Conference*, 458-463.

CHO, J. D., RAJE, S., SARRAFZADEH, M., SRIRAM, M., AND KANG, S. M. 1993. Crosstalk-minimum layer assignment. In *Proceedings of the Custom Integrated Circuits Conference*. 29.7.1–29.7.4.

CONG, J., FANG, J., AND KHOO, K. 2001. DUNE: A multilayer gridless routing system. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 20, 5, 633–646.

DION, J. AND MONIER, L. M. 1995. Contour: A tile-based gridless router. Western Research Laboratory Res. rep. 95/3.

GAO, T. AND LIU, C. L. 1996. Minimum crosstalk channel routing. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 15*, 5, 465–474.

GUO, P. N., CHENG, C. K., AND YOSHIMURA, T. 2001. Floorplanning using a tree representation. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 20*, 2, 281–289.

HE, L. AND XU, M. 1999. Modeling and layout optimization for on-chip inductive coupling. Tech. rep. ECE-00-1. University of Wiscons in at Madison.

HO, T. Y., CHANG, Y. W., CHEN, S. J., AND LEE, D. T. 2005. Crosstalk-and performance-driven multilevel full-chip routing. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 24*, 6, 869–878.

KAO, W. C. AND PARNG, T. M. 1995. Cross point assignment with global rerouting for general-architecture designs. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 14*, 3, 337–348.

KASTNER, R., BOZOGZADEH, E., AND SARRAFZADEH, M. 2000. Predictable routing. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 110–113.

KUO, Y. S., CHERN, T. C., AND SHIH, W. K. 1988. Fast algorithm for optimal layer assignment. In *Proceedings of the 25th ACM/IEEE Conference on Design Automation*, 554–559.

LI, Y. L., CHEN, X. Y., AND LIN, Z. D. 2007. NEMO: A new implicit connection graph-based gridless router with multi-layer planes and pseudo-tile propagation. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 26*, 4, 705–718.

MARGARINO, A., ROMANO, A., GLORIA, A. DE, CURATELLI, F., AND ANTOGNETTI, P. 1987. A tile-expansion router. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 6*, 507–517.

PAN, M. AND CHU, C. 2006. FastRoute: A step to integrate global routing into placement. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 464–471.

SAIT, S. M. AND YOUSSEF, H. 1999. *VLSI Physical Design Automation*. World Scientific Publishing.

SAPATNEKAR, S. S. 2000. A timing model incorporating the effect of crosstalk on delay and its application to optimal channel routing. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 19*, 5, 550–559.

SHI, C. J. R. 1997. Solving constrained via minimization by compact linear programming. In *Proceedings of the Asia and South Pacific Design Automation Conference*. 635–640.

TSENG, H. P., SHEFFER L., AND SECHEN, C. 2001. Timing- and crosstalk-driven area routing. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 20*, 4, 528–544.

TU, S. W., SHEN, W. Z., CHANG, Y. W., CHEN, T. C., AND JOU, J. Y. 2003. Inductance modeling for on-chip interconnects. *Int. J. Analog Integr. Circ. Sig. Proce. 35*, 1, 65–78.

VITTAL, A. AND MAREK-SADOWSKA, M. 1997. Crosstalk reduction for VLSI. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 16*, 4, 290–298.

WU, D., HU, J., ZHAO, M., AND MAHAPATRA, R. 2005. Timing driven track routing considering coupling capacitance. In *Proceedings of the Asia and South Pacific Design Automation Conference*. 1156–1159.

XUE, T., KUH, E. S., AND WANG, D. 1996. Post global routing crosstalk risk estimation and reduction. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 302–309.

ZHOU, H. AND WONG, D. F. 1999. Global routing with crosstalk constraint. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 18*, 11, 1683–1688.


Received March 2009; revised September 2009, March 2010; accepted July 2010