# Multilevel Full-Chip Gridless Routing With Applications to Optical-Proximity Correction

Tai-Chen Chen, *Student Member, IEEE*, and Yao-Wen Chang, *Member, IEEE*

***Abstract*—To handle modern routing with nanometer effects, we need to consider designs with variable wire/via widths and spacings, for which gridless-routing approaches are desirable due to its great flexibility. In this paper, we introduce a gridless-routing model that can obtain design-rule-correct paths and avoid redundant wires. Besides, we propose an enhanced model for the gridless-routing model to reduce the solution space and the runtime. Based on the enhanced gridless-routing model, we present the first multilevel full-chip gridless detailed router (called MGR). The router integrates global routing, detailed routing, and congestion estimation together at each level of multilevel routing. It can handle designs with nonuniform wire/via widths and spacings and consider routability and optical-proximity correction. Experimental results show that MGR achieves the best routing solutions in smaller running times than previous works, based on a set of commonly used benchmarks (with uniform and nonuniform wire widths) and a set of real industrial benchmarks (with a versatile set of design rules).**

***Index Terms*—Design for manufacturing (DFM), gridless routing, multilevel optimization, optical-proximity correction (OPC), physical design, routing.**

## I. Introduction

RESEARCH in very large-scale integration (VLSI) routing has received much attention in the literature. Routing is typically a very complex combinatorial problem. In order to make it manageable, the routing problem is usually solved using the two-stage approach of global routing followed by detailed routing. Global routing first partitions the routing area into tiles and decides tile-to-tile paths for all nets, while detailed routing assigns actual tracks and vias for nets. Many routing algorithms adopt a flat framework of finding paths for all nets. Those algorithms can be classified into sequential and concurrent approaches. Early sequential-routing algorithms include maze-searching [28], [42] and line-searching approaches [18], which route net-by-net. Most concurrent algorithms apply network-flow or linear-assignment formulation [2], [37] to route a set of nets at one time.

Manuscript received October 8, 2005; revised April 1, 2006. This work was supported in part by SpringSoft, Inc., and the National Science Council of Taiwan, under Grants NSC 93-2215-E-002-009 and NSC 93-2215-E-002-029. This paper was presented in part at the 2005 ACM/IEEE Asia and South Pacific Design Automation Conference [7]. This paper was recommended by Associate Editor T. Yoshimura.

T.-C. Chen is with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: tcchen@eda.ee.ntu.edu.tw).

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

Digital Object Identifier 10.1109/TCAD.2006.884492

The major problem of the flat frameworks lies in their scalability for handling larger designs. As technology advances, technology nodes are getting smaller and circuit sizes are getting larger. To cope with the increasing complexity, researchers proposed to use hierarchical approaches to handle the problem: Marek-Sadowska in [35] proposed a hierarchical global router based on linear assignment; Heisterman and Lengauer in [17] presented a hierarchical integer-linear programming approach for global routing; Wang and Kuh in [44] proposed a hierarchical $(\alpha, \beta)^*$ algorithm for timing-driven multilayer MCM/IC routing; Chang *et al.* in [5] applied linear assignment to develop a hierarchical, concurrent global, and detailed router for field programmable gate arrays.

The two-level hierarchical routing framework, however, is still limited in handling the dramatically growing complexity and maintaining high-solution quality at the same time in current and future IC designs, which may contain billions of transistors in a single chip. As pointed out in [12], for a 0.07-$\mu$m 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 larger IC designs.

### A. Multilevel Routing

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 [3], HPM [10], and hMETIS [26], the multilevel floorplanner/placer $MB^* - \mathrm{tree}$ [29], and the multilevel placers mPL [4] and APlace [24], [25] all show the promise of the multilevel framework for large-scale circuit partitioning, floorplanning, and placement.

A framework similar to multilevel routing was presented in [16], [30], and [34]. Lin *et al.* in [30] and Hayashi and Tsukiyama in [16] presented hybrid hierarchical global routers for multilayer VLSIs, in which both bottom–up (coarsening) and top–down (uncoarsening) techniques were used for global

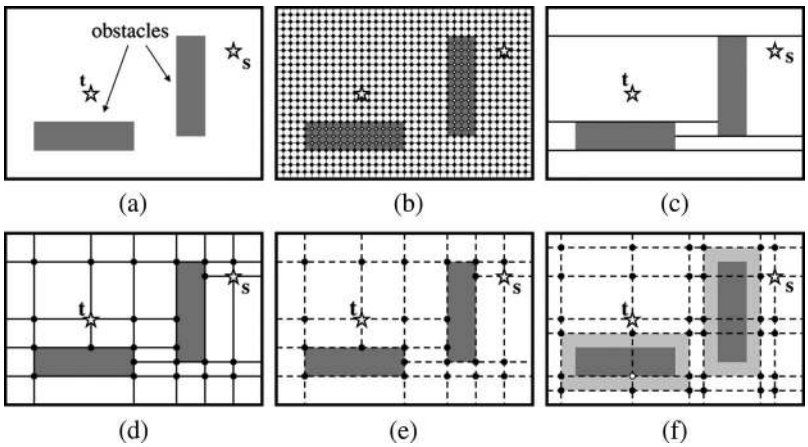


Fig. 1. (a) Routing example. $s$ and $t$ are the source and target of a routing wire, respectively. The two rectangles represent obstacles. (b) Uniform-grid model. The black and white circles denote the routable and unroutable nodes, respectively. (c) Tile-based model. (d) Nonuniform-grid model. (e) Implicit nonuniform-grid model. (f) Nonuniform-grid graph model. The gray areas denote the obstacle zones constructed by expanding obstacles according to design rules.

routing. Marek-Sadowska in [34] proposed a global router based on the outer most loop approach. The approach is similar to the coarsening stage of multilevel routing. Cong *et al.* in [12] proposed a pioneering routability-driven multilevel global-routing approach for large-scale full-chip routing. Cong *et al.* later proposed an enhanced multilevel-routing system, named MARS, which incorporates resource reservation, a graph-based Steiner tree heuristic, and a history-based multiiteration scheme to improve the quality of the multilevel-routing algorithm in [13] and [14]. Their final results of the multilevel global routing are tile-to-tile paths for all nets. The results are then fed into a nonmultilevel gridless detailed router, called DUNE [9], [11], to find the exact connection for each net (therefore, MARS is in fact a multilevel global router, but not a detailed router). Lin and Chang in [6] and [31] proposed a multilevel approach for full-chip grid-based routing, which considers both routability and performance. This framework integrates grid-based 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 for grid-based routing. Recently, Ho *et al.* in [19]–[21] presented another multilevel framework for full-chip grid-based routing considering crosstalk and antenna effects, respectively. The framework incorporates an intermediate stage of layer/track assignment between the coarsening stage and the uncoarsening stage. The coarsening stage performs only global routing while global and detailed routing are integrated together at the uncoarsening stage.

### B. Gridless Detailed Routing

In the detailed-routing stage, seeking design-rule-correct paths in the routing region is a major concern. Traditional detailed routings use uniform-grid models to simplify the problem, as shown in Fig. 1(b). However, uniform-grid models need very fine grids to handle nonuniform wire/via widths and spacings, implying larger searching time and storage requirements. Therefore, the grid-based approach is not effective to handle modern routing problems with nanometer electrical effects, such as optical-proximity correction (OPC) and phase-shift mask. To cope with these nanometer electrical effects, gridless-routing models are desirable due to their great flexibility.

Several gridless-detailed-routing models have been investigated and can be classified into two types: tile-based models [32], [33], [36], [40], [43], [47] and connection-graph models [8], [23], [38], [45], [48]. As shown in Fig. 1(c), the tile-based model partitions the routing region into tiles along the boundaries of obstacles and represents the routing region by a data structure such as corner stitching [39]. Therefore, the routing problem is reduced to searching a tile-to-tile path among these tiles. Although searching a tile-to-tile path is fast, manipulating tiles such as insertion and deletion is a complex process. Furthermore, it needs postprocessing to obtain a final design-rule-correct route for the tile-to-tile path. Moreover, it is not easy to apply the tile-based models to multilayer routing with more complex design rules [12].

As shown in Fig. 1(d), Ohtsuki in [38] proposed a nonuniform-grid model, which was constructed by extending lines through the boundaries of all obstacles until they intersect with other obstacles or boundaries of the routing region. Because the preconstruction and representation of the nonuniform-grid model are costly, previous works [8], [23], [45] tried to simplify the nonuniform-grid model by various techniques. However, those techniques are still costly for large-scale designs. As shown in Fig. 1(e), Zheng *et al.* in [48] presented an implicit nonuniform-grid model, which does not construct a connection graph explicitly and characterizes the search space in an on-the-fly fashion during routing. Schiele *et al.* in [41] constructed the obstacle zones from the obstacles by taking design rules into account. The area outside of the obstacle zones is available for placing the center lines of wires and midpoints of contacts. As shown in Fig. 1(f), Cong *et al.* in [9] and [11] integrated the concepts of obstacles zones [41] and the implicit nonuniform-grid model [48] to build their nonuniform-grid-graph (NGG) model. Although the NGG model has the advantages of fast implicit connection graph construction, routing based on the NGG model may incur design-rule-incorrect paths. Furthermore, routing based on the

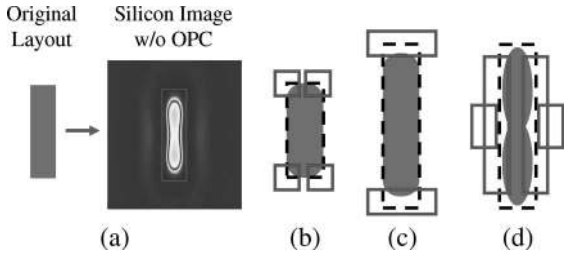


Fig. 2. (a) Optical-proximity effects (courtesy of Synopsys). Three major OPC techniques: (b) Serif; (c) Hammerhead; and (d) Line Biasing.

NGG model may result in redundant wires even using point-to-path maze routing.

### C. OPC Technologies

As the process technology continues to advance, the minimum feature size of circuit patterns becomes significantly smaller than the lithographic wavelength. As a result, it is very hard to obtain the exact image we desire on the wafer. Resolution enhancement techniques, such as OPC, are needed to achieve acceptable process accuracy.

Applying optics simulation, we can clearly see this variation as shown in Fig. 2(a). These variations can be classified into mainly three types: corner rounding, line-end shortening, and linewidth shrinking, as shown in Fig. 2(b)–(d). (Here, a line is a horizontal/vertical segment of a net or a via.) For each type of variations, we can add pattern features to compensate for the distortions and acquire what we really need on the resist. We can add serifs at corners to make the angles sharper, add hammerheads at line ends to compensate for their shortenings, and add line biasings along line sides to compensate for their shrinkings.

OPC might incur a much larger number of pattern features, implying larger memory requirements to record these features and higher mask costs. More than a five times increases in data volume and several days of additional CPU runtime are common side effects of OPC insertion in current designs [15]. If a router can generate the configurations that require fewer shots to OPC, we can reduce the data volume for OPC. Therefore, it is desired to consider the optical effects to reduce the number of pattern features.

As a relatively new issue, there is not much work in the literature on routing with the OPC consideration. Huang and Wong in [22] presented a pioneering work on OPC-friendly maze routing based on the Lagrangian relation formulation. However, the router is grid based and considers only two-pin connections. Furthermore, it uses the flat framework and, thus, cannot handle the problem sizes of thousands of nets well. Recently, Wu *et al.* in [46] presented an enhanced maze routing to solve two OPC-aware maze-routing problems. However, the goals of both problems are to find the single-routing path of the $k$th net when a routing region with $k-1$ routed paths of two-pin nets are given. Although their method can find a routing path for the $k$th net under different constraints and objective functions, it does not consider multipin nets and the net ordering problems.

### D. Our Contribution

In this paper, we propose the first multilevel full-chip gridless detailed router. The four main features of the proposed method are as follows: 1) A gridless-routing model and its enhanced model that can obtain design-rule-correct paths and avoid redundant wires; 2) the first multilevel gridless router that integrates gridless global and detailed routings; 3) a multilevel gridless router that can handle designs with nonuniform wire/via widths and spacing; and 4) a routability-driven and OPC-aware multilevel gridless router that can optimize routing-completion rates and reduce OPC-pattern feature requirements.

Experimental results show that our multilevel gridless router (called MGR) achieves 100% routing-completion rates with less running times than previous works based on a set of commonly used Microelectronics Center of North Carolina (MCNC) benchmarks. Furthermore, MGR can handle designs with nonuniform wire widths well and obtain better routing solutions (still maintain 100% routing completion for all circuits) than the state-of-the-art multilevel gridless-routing system (multilevel gridless global routing + flat gridless detailed routing) [14]. In particular, MGR is the first router to complete the routing for the set of commonly used MCNC benchmarks of nonuniform wire sizes and to route the real industrial Faraday benchmarks with a versatile set of design rules. Moreover, our OPC-aware MGR archives an average reduction of 9% pattern features and still maintains 100% routability for the 11 MCNC-benchmark circuits.

The rest of this paper is organized as follows. Section II presents the global, detailed, and multilevel-routing models. Section III presents our framework for routability-driven and OPC-aware routings. Experimental results are shown in Section IV. Finally, we give concluding remarks in Section V.

## II. Preliminaries

Routing in modern ICs is a very complex process, and we can hardly obtain high-quality solutions directly. Therefore, the routing problem is usually solved using the two-stage approach of global routing followed by detailed routing. Global routing first partitions the routing area into tiles and decides tile-to-tile paths for all nets, while detailed routing assigns actual tracks and vias for nets.

### A. Modeling of Global Routing

Our global-routing algorithm is based on a graph-search technique guided by the congestion associated with routing regions and topologies. The router assigns higher costs to route nets through congested areas to balance the net distribution among routing regions.

Before we can apply the graph-search technique to multilevel routing, we first need to model the routing resource as a graph such that the graph topology can represent the chip structure. Fig. 3 illustrates the graph modeling. For the modeling, we first partition a chip into an array of rectangular subregions. These subregions are called global-routing cells (GRCs). A node in the routing graph represents a GRC in the chip, and an edge

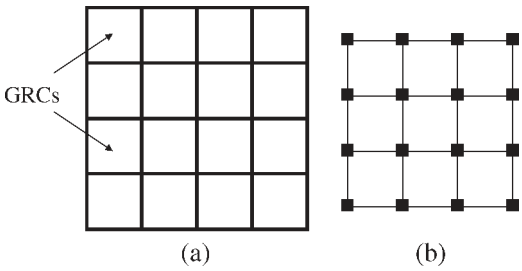


Fig. 3. Modeling of global routing: (a) partitioned layout and (b) routing graph.

denotes the boundary between two adjacent GRCs. Each edge is assigned a capacity according to the width/height of a GRC. The routing 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 GRC-to-GRC paths for all nets on $G_0$ to guide the detailed router. The goal of global routing is to route as many nets as possible, while meeting the capacity constraint of each edge and any other constraint, if specified. Note that, because of the gridless nature of our routing problem, the cost of routing a net is associated with the wire width and spacing.

### B. Modeling of Detailed Routing

Seeking design-rule-correct and high-quality paths are two major concerns for detailed routing, which rely greatly on an appropriate detailed routing model. In the following, we first review the recently proposed NGG model and point out its deficiencies of incurring design-rule-incorrect paths and redundant wires. We then present the implicit-triple-line-graph (ITLG) model to remedy those deficiencies. We further propose the enhanced-ITLG (EITLG) model to enhance the ITLG model by reducing its induced solution space and, thus, running time.

*1) NGG Model:* Cong *et al.* in [9] and [11] integrated the concepts of obstacle zones [41] and the implicit nonuniform-grid model [48] to build their NGG model. For each obstacle, its obstacle zone is constructed by expanding the obstacle for a range, which is the sum of the obstacle spacing and the half width of the routing wire. As shown in Fig. 4(a), the expanded range (gray area) is the sum of $d_s$ and $w_{i/2}$, where $d_s$ and $w_i$ are the obstacle spacing to satisfy the design rules and the width of the routing wire, respectively. With the boundaries of each obstacle zone, two $x$-coordinates (the left and right boundaries) and two $y$-coordinates (the top and bottom boundaries) are obtained. The $x$-coordinates and $y$-coordinates of all obstacle zones and the source $s$ and target $t$ of the routing wire are stored into two sets, $\text{ICG}_x$ and $\text{ICG}_y$, separately. Based on $\text{ICG}_x$ and $\text{ICG}_y$, an implicit connection graph is constructed as shown in Fig. 4(b). A vertical (horizontal) dashed lines in the implicit connection graph is generated through each $x$-coordinate ($y$-coordinate) in $\text{ICG}_x$ ($\text{ICG}_y$). A node in the implicit connection graph denotes an intersection of a horizontal and a vertical dashed lines. There are two types of nodes: routable nodes and unroutable nodes. A routable node allows a routing path to pass through it without violating the design rules; it is unroutable, otherwise. As shown in Fig. 4(b)–(e), the respective black and white circles are the routable and unroutable nodes. To seek a design-rule-correct path from the source $s$ to the target $t$, therefore, we only need to check if there exists a feasible path along which all nodes are routable.

The NGG model has the advantage of fast implicit connection-graph construction. The NGG model performs routing by checking if a feasible path from the source to the target exists. Even though all nodes along this path are routable nodes, it may still incur design-rule-incorrect paths. Fig. 4(b) shows the implicit connection graph constructed by the NGG model for the simple example of Fig. 4(a). Since all nodes are routable nodes (black circles), a shortest path that connects two routable nodes between $s$ and $t$ can be found [see Fig. 4(c)]. The resulting path is not a legal solution, however, since the obstacle and the routing wire do not have enough spacings to satisfy the design rules.

For multiterminal nets, Cong *et al.* in [14] used the NGG model to construct an implicit connection graph and applied the $A^*$ point-to-path maze-searching algorithm to find a path from the source to the target. Nevertheless, the resulting path may incur redundant wires, implying that we need postprocessing to remove the redundant wires and more runtime to obtain a final solution. As the routing example shown in Fig. 5(a), the target $t$ connects with the prerouted wire. Using the implicit connection graph constructed by the NGG model will result in the final path shown in Fig. 5(c), which incurs a redundant wire.

*2) ITLG Model:* To find a design-rule-correct path and avoid redundant wires, we introduce the ITLG model. For each obstacle, as usual, its obstacle zone is constructed by expanding the obstacle for a range, which is the sum of the obstacle spacing and the half width of the routing wire. In addition to the coordinates along the boundaries of an obstacle zone, the ITLG model also stores the coordinate of the center of the obstacle zone into $\text{ICG}_x$ and $\text{ICG}_y$. For each obstacle zone, therefore, three $x$-coordinates (the left boundary, the right boundary, and the center) and three $y$-coordinates (the top boundary, the bottom boundary, and the center) are obtained. As mentioned in Section II-B1, the $x$-coordinates and $y$-coordinates of all obstacle zones and the source $s$ and target $t$ of the routing wire are stored into two sets, $\text{ICG}_x$ and $\text{ICG}_y$, separately. Based on $\text{ICG}_x$ and $\text{ICG}_y$, we can obtain an implicit connection graph with vertical and horizontal dashed lines, as shown in Fig. 4(d). To find a design-rule-correct path from the source $s$ to the target $t$, therefore, we only need to check if there exists a feasible path along which all nodes are routable.

For each obstacle zone, we refer to a perpendicular line (center line) as the dashed line that passes the center of the obstacle zone and perpendicular (parallel) to the routing direction of the obstacle zone. The routing direction of an obstacle zone is horizontal (vertical), if the obstacle zone is located in a horizontal (vertical) routing layer. The ITLG model uses the perpendicular and center lines to find a design-rule-correct path and to avoid redundant wires, respectively. We elaborate on the use of the two lines as follows: 1) **Perpendicular line**: The perpendicular lines are used to avoid design-rule-incorrect paths. Adding the perpendicular line of an obstacle zone introduces intersections of the perpendicular line and the lines, which are orthogonal to the perpendicular line in the implicit connection graph. Since

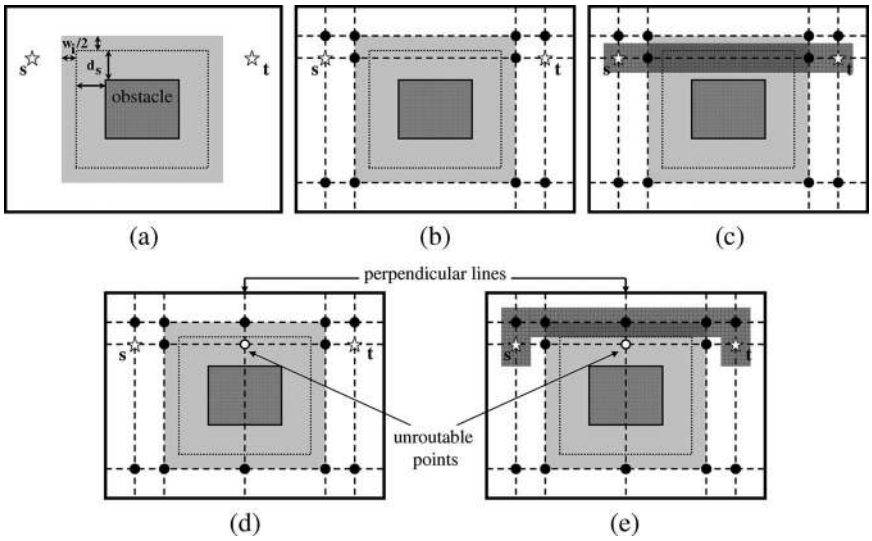


Fig. 4. (a) Routing example. The gray area denotes the obstacle zone constructed by expanding a range, which is the sum of the wire spacing and the half width of the routing wire. $d_s$ and $w_i$ are wire/via spacing that satisfies the design rules and the width of the routing wire, respectively. $s$ and $t$ are the source and target of the routing wire, respectively. (b) Implicit connection graph constructed by the NGG model. The black circles denote the routable nodes. (c) Design-rule-incorrect path, for which the obstacle and the routing wire do not have enough spacings. (d) Implicit connection graph constructed by our ITLG model. The black and white circles denote the routable and unroutable nodes, respectively. (e) Design-rule-correct path found through five routable nodes.

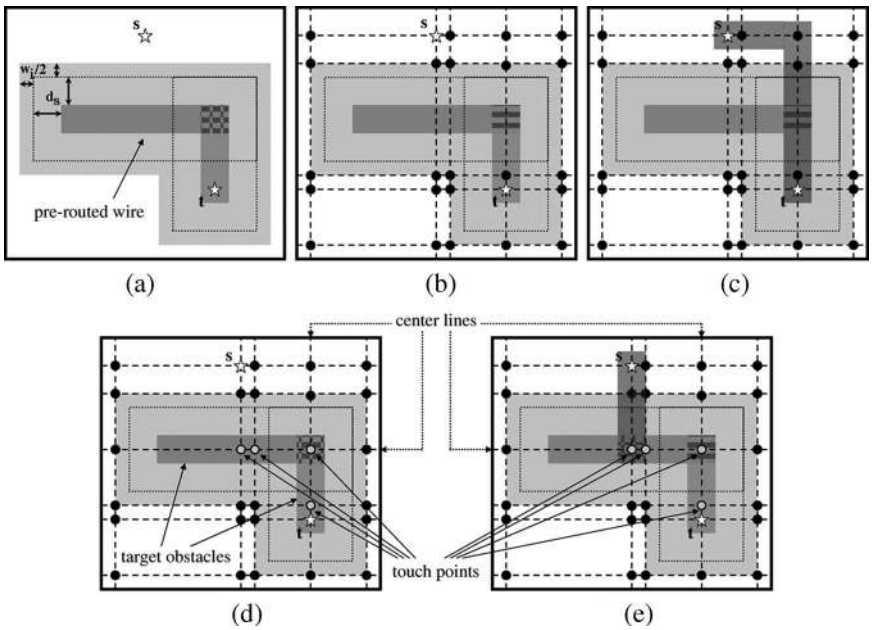


Fig. 5. (a) Routing example. The gray areas denote the obstacle zones constructed by expanding a range, which is the sum of the wire spacing and the half width of the routing wire. The horizontal (vertical) edge of the prerouted wire is in a horizontal (vertical) routing layer. The target of the routing wire $t$ connects with the prerouted wire. $d_s$ and $w_i$ are the obstacle spacing that satisfies the design rules and the width of the routing wire, respectively. $s$ and $t$ are the source and target of the routing wire, respectively. (b) Implicit connection graph constructed by the NGG model. The black circles denote the routable nodes. Since $t$ connects with the prerouted wire, the nodes in the obstacle are routable nodes. (c) Path is found through four routable nodes and incurs a redundant wire between the prerouted wire and the routing wire. (d) Implicit connection graph constructed by our ITLG model. The gray circles denote the touch nodes. (e) Path is found through two routable nodes and incurs no redundant wire between the prerouted wire and the routing wire.

the nodes (intersections) are located inside the obstacle zone, they are unroutable nodes. These unroutable nodes can be used to avoid a path from crossing the obstacle zone directly. As shown in Fig. 4(d), the white circle is introduced by the perpendicular line and is an unroutable node. With the unroutable node in mind, we can find the design-rule-correct, detour path shown in Fig. 4(e) and avoid the design-rule-incorrect path shown in Fig. 4(c). 2) **Center line**: The center lines are used to avoid redundant wires. Adding the center line of an obstacle zone introduces intersections of the center line and the lines, which are orthogonal to the center line in the implicit connection graph. Nodes (intersections) located inside target obstacles are touch nodes, which are also routable nodes. Target obstacles denote obstacles (prerouted wires), which are connected with the target of the routing wire. If the source of the routing wire connects with one of these touch nodes, the source and the target are connected. Therefore, we can find a path from the source to one of the touch nodes to avoid redundant wires. As

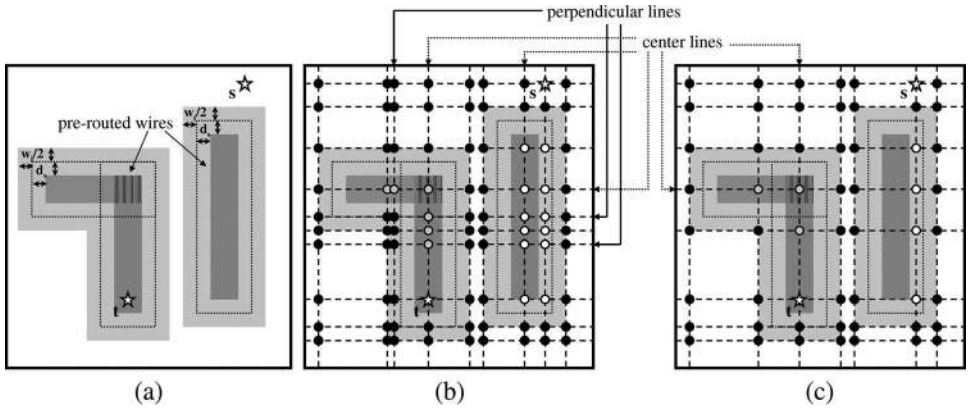


Fig. 6. (a) Routing example. The gray areas denote the obstacle zones constructed by expanding a range which is the sum of the wire spacing and the half width of the routing wire. The target of the routing wire connects with the prerouted wire in the left side. $d_s$ and $w_i$ are wire/via spacing that satisfies the design rules and the width of the routing wire, respectively. $s$ and $t$ are the respective source and target of the routing wire. (b) Implicit connection graph constructed by our ITLG model. The total number of nodes is $9 \times 10$. (c) Implicit connection graph constructed by our EITLG model. The total number of nodes is reduced to $7 \times 8$. The black, white, and gray circles denote the routable, unroutable, and touch nodes, respectively. Because the target of the routing wire connects with the prerouted wire in the left side, the nodes located above the obstacle zones in the left (right) side are routable (unroutable) nodes.

shown in Fig. 5(d), touch nodes are located inside the prerouted wire. Therefore, we can find a shorter path with nonredundant wires shown in Fig. 5(e).

*3) EITLG Model:* Although the ITLG model can avoid design-rule-incorrect paths and redundant wires, the induced solution space (the total number of nodes) of the ITLG model may be more than twice of that of the NGG model, implying larger runtime and storage requirements. Therefore, we shall propose an enhanced model, called the EITLG model, to reduce the solution space of the ITLG model.

We show how to reduce the solution space of the ITLG model as follows: 1) **Perpendicular line**: To avoid a design-rule-incorrect path, the ITLG model uses the perpendicular lines of all obstacle zones to identify unroutable nodes located inside the zones to avoid a path from crossing the zone directly. If there exists a line parallel to the perpendicular line of an obstacle zone and passing through the zone, this line can introduce unroutable nodes inside the zone, too. Since both of this line and the perpendicular line can introduce unroutable nodes inside the obstacle zone, we do not need to construct the perpendicular line. As shown in Fig. 6, each obstacle zone contains lines perpendicular to the routing direction of the zone and passing through the zone. Therefore, the EITLG model does not construct the perpendicular line for each obstacle zone, as shown in Fig. 6(c), reducing significant solution space for detailed routing. 2) **Center line**: To avoid redundant wires, the ITLG model uses center lines of all obstacle zones to generate nodes. Since only the nodes located inside target obstacles are touch nodes, we only need to construct the center lines of the target obstacles (obstacle zones). As shown in Fig. 6, only the two obstacles in the left side connect with the target. Therefore, the EITLG model does not construct the center line of the obstacle in the right side of Fig. 6(c).

Algorithm ICG construction, shown in Fig. 7, constructs $\text{ICG}_x$ and $\text{ICG}_y$ for routing a wire from the source $s$ to the target $t$ based on the EITLG model. The time complexity of this algorithm is $O(n \lg n)$ by implementing the two sets $\text{ICG}_x$ and $\text{ICG}_y$ with two sorted arrays, where $n$ is the total number of obstacle zones.

**Algorithm:** ICG-Construction($Z$, $s$, $t$)
**Input:** $Z$ - the set of obstacle zones $\{Z_1, Z_2, ..., Z_n\}$;
$s$ - the source of the routing wire;
$t$ - the target of the routing wire;
$b_i^l$ - the $x$-coordinates of the left boundary of $Z_i$;
$b_i^r$ - the $x$-coordinates of the right boundary of $Z_i$;
$b_i^b$ - the $y$-coordinates of the bottom boundary of $Z_i$;
$b_i^t$ - the $y$-coordinates of the top boundary of $Z_i$;
$x_i^c$ - the $x$-coordinate of the center of $Z_i$;
$y_i^c$ - the $y$-coordinate of the center of $Z_i$;
$x^s$ - the $x$-coordinate of $s$;
$y^s$ - the $y$-coordinate of $s$;
$x^t$ - the $x$-coordinate of $t$;
$y^t$ - the $y$-coordinate of $t$.
**Output:** $ICG_x$ and $ICG_y$.
1 $ICG_x \leftarrow \{x^s, x^t\}$;
2 $ICG_y \leftarrow \{y^s, y^t\}$;
3 **for** $i \leftarrow 1$ to $n$ **do**
4 $ICG_x \leftarrow ICG_x \cup \{b_i^l, b_i^r\}$;
5 $ICG_y \leftarrow ICG_y \cup \{b_i^b, b_i^t\}$;
6 **for** $i \leftarrow 1$ to $n$ **do**
7 **if** $Z_i$ is in a horizontal routing layer
8 **if** ($\forall x \in ICG_x$ s.t. $x \leq b_i^l$ or $x \geq b_i^r$)
9 $ICG_x \leftarrow ICG_x \cup \{x_i^c\}$;
10 **if** ($Z_i$ and $t$ are connected)
11 $ICG_y \leftarrow ICG_y \cup \{y_i^c\}$;
12 **else** /* $Z_i$ is in a vertical routing layer */
13 **if** ($\forall y \in ICG_y$ s.t. $y \leq b_i^b$ or $y \geq b_i^t$)
14 $ICG_y \leftarrow ICG_y \cup \{y_i^c\}$;
15 **if** ($Z_i$ and $t$ are connected)
16 $ICG_x \leftarrow ICG_x \cup \{x_i^c\}$;
17 **return** $ICG_x$ and $ICG_y$;

Fig. 7. Algorithm to construct $\text{ICG}_x$ and $\text{ICG}_y$ for routing a wire from the source $s$ to the target $t$ based on the EITLG model.

### C. Modeling of Multilevel Routing

Fig. 8 shows our multilevel framework. As illustrated in Fig. 8, $G_0$ 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 tile) and, then, the detailed router is used to determine the exact wiring. The congestion estimation is performed after the detailed routing finishes routing a net. After the global routing,

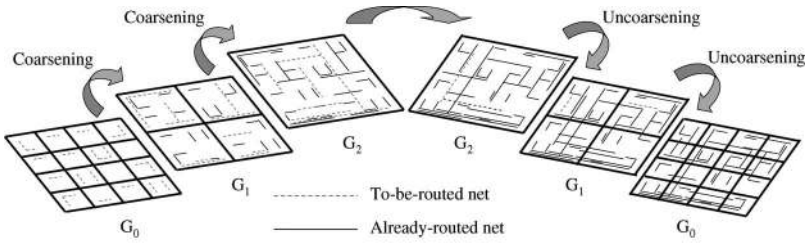


Fig. 8. Multilevel-framework flow.

detailed routing, and congestion estimation are performed, we merge four adjacent tiles of $G_0$ into a larger tile for use at the next level (i.e., level 1 here). Coarsening continues until the number of tiles at a level, say the $k$th level, is below a threshold. After finishing coarsening, the uncoarsening stage tries to refine the routing solution starting from the last level $k$ where coarsening stops. During uncoarsening, the unroutable nets during coarsening are considered, and maze routing and rip-up and reroute are performed to refine the routing solution. Then, we proceed to the next level (level $k-1$) of uncoarsening by expanding each tiles to four finer tiles. The process continues up to level 0 when the final routing solution is obtained.

## III. Multilevel Routing Framework

Although our multilevel routing framework is inspired by the work [6] and [31], our routing model is totally different from that of [6] and [31]. The works [6] and [31] are for grid-based routing and, thus, they can apply a graph-searching technique (e.g., breadth-first search or depth-first search) on predefined grids. However, this paper is for gridless routing. Since the solution space of gridless routing is significantly larger than that of grid-based routing, we apply a different routing model for gridless routing, as discussed in Section II-B. Besides, our congestion estimation is significantly different from that of [6] and [31].

MGR tends to route wider nets first, since a wider net requires more routing resource. Beside, MGR tends to route shorter nets first, since we route local nets at each level of coarsening. It is obvious that the local nets at the lower level (say, level 0) are usually shorter than those at the higher level (say, level $k$). Naturally, a shorter net enjoys less freedom while searching for a path to route it. This fact holds even during rip-up and reroute. Thus, this observation implicitly suggests that a shorter net has a higher priority than a longer net as far as routability is concerned. Kastner *et al.* in [27] also suggested this finding. Although this net ordering scheme may not be the optimal solution for some routing problems (for example, when timing is considered, routing the most critical net first often leads to better timing performance), it is still a reasonable alternative.

### A. Multilevel Routing

In the following, we present our framework for multilevel-gridless routing and summarize it in Fig. 9.

```
Algorithm: Multilevel-Gridless-Routing(G, N)
Input:  G - a placed layout;
        N - netlist of multi-terminal nets.
Output: routing solutions for N on G.
1   Partition G;
2   For each net n ∈ N
3       Construct an MST;
4       Decompose the MST into 2-pin connections;
5   // Coarsening Stage
6   For each level at the coarsening stage
7       For each connection that belongs to this level
8           Perform global pattern routing;
9           Perform detailed routing;
10          Perform congestion estimation;
11  // Uncoarsening Stage
12  For each level at the uncoarsening stage
13      For each failed connection at the coarsening stage
14          Perform global maze routing;
15          Perform detailed routing;
16          Perform congestion estimation;
17  return the routing layout;
```

Fig. 9. Multilevel gridless-routing algorithm.

Given a netlist, we first run the minimum spanning tree (MST) algorithm to construct the topology for each net and, then, decompose each net into two-pin connections, with each connection corresponding to an edge of the MST. Our multilevel framework starts from coarsening the finest tiles of level 0. At each level, tiles are processed one by one, and only local nets (connections) are routed. At each level, the two-stage routing approach of global routing followed by detailed routing is applied.

The global routing is based on the approach used in the Pattern Router [27] and first routes local nets (connections) on the tiles of level 0. Let the multilevel routing graph of level $i$ be $G_i = (V_i, E_i)$. Let $R_e = \{e \in E_i \mid e$ is the edge chosen for routing$\}$. We apply the cost function $\alpha : E_i \rightarrow \Re$ to guide the routing

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

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

$$c_e = \frac{d_e}{p_e}$$

where $d_e$ and $p_e$ are the density and capacity associated with $e$, respectively. Pattern routing uses an L- or Z-shaped route to make the connection, which gives the shortest path length

between two points. Therefore, the wire length is minimum. We measure the routing congestion based on the channel density defined by the sum of wire spacing and wire width for gridless routing (note that the definition is different from the case in grid-based routing, for which channel density is defined as the maximum number of parallel nets passing through a routing channel). If pattern routing fails, we give up routing the connection. We refer to a failed net (failed connection) as that causes an overflow. The failed nets (connections) will be reconsidered (refined) at the uncoarsening stage.

After the global routing is completed, we perform detailed routing with the guidance of the global-routing results and find a real path in the chip. Our detailed router is based on Dijkstra's shortest path algorithm and supports local refinement. After the detailed routing finishes routing a net, the channel density associated with an edge of a multilevel graph is updated accordingly. This is called congestion estimation. There are at least two advantages by using this approach. First, routing-congestion estimation is more accurate than that performing global routing alone, since we can precisely evaluate the routing region. Second, we can obtain a good initial solution for the following refinement very effectively, since pattern routing enjoys very low time complexity and uses fewer routing resources due to its simple L- and Z-shaped routing patterns.

The uncoarsening stage starts to refine each local failed net (connection), left from the coarsening stage. The global router is now changed to the maze router with the same cost function in the coarsening stage. Uncoarsening continues until the first level $G_0$ is reached and the final solution is found. Note that the global maze routing, here, serves as an elaborate rip-up and reroute processor, in contrast to the simple L- and Z-shaped routing during coarsening (for rip-up and reroute in our multilevel-routing algorithm, we mean the maze routing at the uncoarsening stage; it is only applied to global routing for better efficiency and quality tradeoff). This two-stage approach of global and local refinement of detailed routing gives our overall refinement scheme.

### B. OPC-Aware Multilevel Routing

In modern nanometer-process technologies, such as 90-nm technology and beyond, most of the metal layers need OPC to control the linewidth and length variations. Considering OPC in the routing stage, we can maximize the effects of the correction and, thus, reduce the number of OPC-pattern features during masking.

There are generally two major approaches to OPC: model-based and rule-based approaches. The model-based method applies optics simulation to add OPC pattern features to fix the OPC problem. It is typically more accurate, but is much more complicated and time-consuming. In contrast, the rule-based method adds the OPC-pattern features based on some predefined design rules. This approach is inevitably less accurate, but is much simpler and more efficient. Our router adopts the rule-based approach since it is obviously not feasible to incorporate the very time-consuming model-based approach into a multilevel-routing framework.

In the following, we demonstrate how to consider the OPC rules during routing by incorporating a set of major OPC design rules into the cost function of our router. Note that it is not our intention here to elaborate on all OPC design rules or the accuracy of the rules. Though not presented here, nevertheless, it is not hard to incorporate other (more accurate) OPC design rules into the cost function of our router.

*1) OPC Cost Function:* The OPC effect of a line is related to its neighboring configuration. Since the neighboring configuration of a line is not fixed in the routing stage (not all lines are routed), it is very hard to evaluate the OPC cost with unfixed neighboring configuration. Therefore, we propose a combined estimation of actual and estimated OPC cost to calculate the OPC cost for a line considering the routed and unrouted neighboring lines. We define the OPC cost for a line $e$ by

$$\text{cost}(e) = \text{cost}_a(e) + \text{cost}_e(e). \tag{2}$$

The combined cost consists of an actual cost $\text{cost}_a(e)$ (for real neighboring configuration) and an estimated cost $\text{cost}_e(e)$ (for the worst case neighboring configuration). At first, the OPC cost for a line is estimated by the worst case neighboring configuration alone. After a connection is routed successfully, the real neighboring configuration will be updated dynamically. Therefore, our OPC cost is based on the OPC effect incurred by both the already routed nets and the estimated unrouted nets. As routing proceeds, we have more and more accurate OPC effect for routing succeeding nets. We describe how to calculate these two cost as follows: 1) **Actual cost**: We calculate the actual cost for a line based on the OPC effect caused by the neighboring routed lines. In addition, the optical interference is limited within a region of several wavelengths [22], [46]. Therefore, only neighboring routed lines within the effective region (spacing) are considered. Here, the effective region is defined by the foundry. Consequently, we define the actual cost for a line $e$ to be the total number of pattern features as follows:

$$\text{cost}_a(e) = \lfloor l_o/L_L \rfloor + \lfloor w_o/W_L \rfloor \tag{3}$$

where $l_o$ and $w_o$ are the overlapping length and width with the routed neighboring lines within the effective region (spacing), respectively. Here, $L_L$ and $W_L$ (the unit-length and unit-width for adding a pair of line biasings) are parameters related to the process technology and are defined by the foundry. 2) **Estimated cost**: Evaluating the OPC cost for a line without considering the OPC effect caused by the neighboring unrouted lines may be inaccurate. Since we do not know the final layout of the neighboring configuration, we consider the worst case neighboring configuration for the cost estimation. In other words, we assume that a line segment is fully surrounded by adjacent lines. Therefore, the OPC cost for the line is proportional to its length and width. As shown in Fig. 2(b), for a line, we need to add four serifs at the corners to increase the fidelity of images. As the length of a line increases, the ends of the line are shortened. As shown in Fig. 2(c), therefore, we need to add two hammerheads at the line ends for a long line (a line is said to be a long line if its length is longer than or equal to $L_T$, where $L_T$ is the threshold length for a long line

```
1 if (dist(v) ≥ dist(u) + w(u, v))
2   if (v is along the propagation direction of u
        and feature(v) > feature(r) + o(r, v))
3     feature(v) ← feature(r) + o(r, v);
4     Record u as the predecessor routing node of v;
5   if (v is not along the propagation direction of u
        and feature(v) > feature(u) + o(u, v))
6     feature(v) ← feature(u) + o(u, v);
7     r ← u;
8     Record u as the predecessor routing node of v;
```

Fig. 10. Algorithm to compute $\mathrm{feature}(v)$.

and is defined by the foundry). Besides, the overlapping length of a line with neighboring lines may increase as the length of the line increases; further, a wider line is easier to be affected by neighboring lines than a narrower one. These phenomena make the sides of a line shrink more seriously. Therefore, as shown in Fig. 2(d), we need some line biasings in the line sides to correct the optical-proximity effects for a line. The total number of line biasings for a line is determined by the length and width of the line. According to the above modeling, we define the estimated cost for a line $e$ whose length and width are $l_e$ and $w_e$, respectively, to be the total number of pattern features as follows:

$$\mathrm{cost}_e(e) = \begin{cases} 4 + f(l_e, w_e), & \text{when } l_e < L_T \\ 6 + f(l_e, w_e), & \text{otherwise} \end{cases}$$

where $f$ is a step function and is defined as follows:

$$f(l_e, w_e) = 2 \times (\lfloor l_e/L_L \rfloor + \lfloor w_e/W_L \rfloor). \tag{4}$$

Therefore, the total OPC cost for a connection is the sum of the OPC costs for lines that belong to the connection.

*2) OPC Cost Minimization:* We apply the following algorithm, called simultaneous path length and OPC cost Minimization (SPOM), to perform Dijkstra's shortest path algorithm to find a shortest path with the minimum number of pattern features. It associates each basic routing node $u$ (a node in an implicit connection graph) with two labels: $\mathrm{dist}(u)$ and $\mathrm{feature}(u)$, where $\mathrm{dist}(u)$ is the distance of the shortest path from source $s$ to $u$ and $\mathrm{feature}(u)$ is the minimum number of pattern features along the shortest path from $s$ to $u$. Initialize $\mathrm{dist}(u) = \infty, \mathrm{feature}(u) = \infty, \forall u \neq s, \mathrm{dist}(s) = 0$, and $\mathrm{feature}(s) = 0$. The computation of label $\mathrm{dist}s$ is the same as original Dijkstra's shortest path algorithm. Let $u$ be a basic routing node on the wavefront and $v$ be a neighboring basic routing node of $u$. The predecessor routing node of $u$ is the region from which the wavefront was propagated for obtaining the minimum $\mathrm{feature}(u)$. The propagation direction of $u$ is the direction from the predecessor routing node of $u$ to $u$. The computation of $\mathrm{feature}(v)$ is shown in Fig. 10, where $w(u, v)$ and $o(u, v)$ are the distance and the number of additional pattern features between nodes $u$ and $v$, respectively. Besides, $r$ denotes that in the last routing node, we insert a via along the shortest path from $s$ to $u$ and is initialized to $s$.

The basic idea is to compare the distance label $\mathrm{dist}s$ first and then compare the pattern-feature number label $\mathrm{feature}s$. The value $\mathrm{feature}(v)$ of a neighboring routing node $v$ with $\mathrm{dist}(v) < \mathrm{dist}(u)$ stays unchanged because the path from $s$ through $u$ to $v$ is not the shortest path between $s$ and $v$. Note that it is possible that there may exist several shortest paths with different numbers of pattern features. It is clear that Algorithm SPOM guarantees to find a shortest path with the minimum number of pattern features, if such a path exists.

## IV. Experimental Results

We implemented MGR in the C++ language on a 1-GHz Sun Blade-2000 workstation with 8-GB memory. Our routing package is available at http://eda.ee.ntu.edu.tw/research.htm. We used two sets of benchmarks, the MCNC benchmarks (provided by the authors of [14]) and the Faraday benchmarks introduced in [1], for our comparative study on routing. The MCNC benchmarks are considered the largest benchmarks commonly used in academia, while the Faraday benchmarks are real industrial designs with many more nets and more complex design rules than the MCNC benchmarks.

Tables I and II list the set of benchmarks. In these tables, "Circuit" gives the names of the circuits, "Size ($\mu$m$^2$)" gives the layout dimensions in square micrometers, "#Layers" denotes the number of routing layers used, "#Nets" gives the total number of nets, "#Connections" gives the number of two-pin connections after net decomposition, "#Pins" gives the number of pins, "Wire/Via Width ($\mu$m)" gives the design rules for wire/via width, and "Wire/Via Spacing ($\mu$m)" gives the design rules for wire/via spacing. Table III lists the design rules for the Faraday benchmarks with six metal layers and five via layers, including widths and spacings.

### A. Comparison of MGR Based on the ITLG and EITLG Models

Table IV gives the comparison of MGR based on the ITLG and EITLG models. In the table, "#Failed Connections" denotes the number of failed connections, "Comp. Rates" gives the routing completion rates, "#Nodes" denotes the total number of nodes, "Mem. (MB)" denotes the storage requirements in megabytes, and "Time (sec)" represents the runtimes in seconds.

The experimental results show that MGR based on the EITLG model is much more efficient. As shown in the table, MGR based on the EITLG model achieves equal routing solutions to MGR based on the ITLG model with 1.90 $\times$ runtime speedup, 4.65 $\times$ reduction in the total number of nodes, and 1.41 $\times$ reduction in the storage requirements.

### B. Multilevel Routing With MCNC Benchmarks

*1) MCNC Benchmarks With Uniform Nets:* Table V compares MGR with the multilevel grid-based router with the routability mode proposed in [6] and [31] (called MR) and the multilevel gridless-routing system (multilevel gridless global routing + flat gridless detailed routing) in [14] (called MARS).

TABLE I
STATISTICS OF THE MCNC BENCHMARKS. NOTE: WE CORRECT SOME "SIZE," "#CONNECTIONS," AND "#PINS" IN THIS TABLE, SINCE THOSE INFORMATION IN [6], [14], AND [31] ARE INCORRECT

| Circuit | Size ($\mu m^2$) | #Layers | #Nets | #Connections | #Pins | Wire/Via Width ($\mu m$) | Wire/Via Spacing ($\mu m$) |
|---|---|---|---|---|---|---|---|
| Mcc1 | 45000×39000 | 4 | 802 | 1693 | 3101 | 25 | 20 |
| Mcc2 | 152400×152400 | 4 | 7118 | 7541 | 25024 | 25 | 20 |
| Struct | 4903×4904 | 3 | 1920 | 3551 | 5471 | 0.6 | 1.2 |
| Primary1 | 7522×4988 | 3 | 904 | 2037 | 2941 | 0.6 | 1.2 |
| Primary2 | 10438×6488 | 3 | 3029 | 8197 | 11226 | 0.6 | 1.2 |
| S5378 | 435×239 | 3 | 1694 | 3124 | 4818 | 0.36 | 0.36 |
| S9234 | 404×225 | 3 | 1486 | 2774 | 4260 | 0.36 | 0.36 |
| S13207 | 660×365 | 3 | 3781 | 6995 | 10776 | 0.36 | 0.36 |
| S15850 | 705×389 | 3 | 4472 | 8321 | 12793 | 0.36 | 0.36 |
| S38417 | 1144×619 | 3 | 11309 | 21035 | 32344 | 0.36 | 0.36 |
| S38584 | 1295×672 | 3 | 14754 | 28177 | 42931 | 0.36 | 0.36 |

TABLE II
STATISTICS OF THE FARADAY BENCHMARKS

| Circuit | Size ($\mu m^2$) | #Layers | #Nets | #Connections | #Pins |
|---|---|---|---|---|---|
| DMA | 408.4×408.4 | 6 | 13256 | 36162 | 73982 |
| DSP1 | 706×706 | 6 | 28447 | 63495 | 144872 |
| DSP2 | 642.8×642.8 | 6 | 28431 | 63386 | 144703 |
| RISC1 | 1003.6×1003.6 | 6 | 34034 | 95106 | 196677 |
| RISC2 | 959.6×959.6 | 6 | 34034 | 95099 | 196670 |

TABLE III
DESIGN RULES OF THE FARADAY BENCHMARKS. LAYERS METAL1, METAL2, METAL3, METAL4, METAL5, AND METAL6 ARE ROUTING LAYERS WHILE LAYERS V1, V2, V3, VL, AND VQ ARE VIA LAYERS (-: VIA LAYERS DO NOT HAVE THOSE INFORMATION)

| Layer | metal1 | V1 | metal2 | V2 | metal3 | V3 | metal4 | VL | metal5 | VQ | metal6 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Layer Type | Routing | Cut | Routing | Cut | Routing | Cut | Routing | Cut | Routing | Cut | Routing |
| Routing Direction | Horizontal | – | Vertical | – | Horizontal | – | Vertical | – | Horizontal | – | Vertical |
| Width ($\mu m$) | 0.16 | 0.32 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.4 | 0.4 | 0.4 | 0.4 |
| Spacing ($\mu m$) | 0.16 | 0.28 | 0.2 | 0.28 | 0.2 | 0.28 | 0.2 | 0.56 | 0.4 | 0.56 | 0.4 |

TABLE IV
COMPARISON OF (A) MGR BASED ON THE ITLG MODEL AND (B) MGR BASED ON THE EITLG MODEL

| Circuit | (A) MGR based on the ITLG model | | | | | (B) MGR based on the EITLG model | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | #Failed Connections | Comp. Rates | #Nodes | Mem. (MB) | Time (sec) | #Failed Connections | Comp. Rates | #Nodes | Mem. (MB) | Time (sec) |
| Mcc1 | 0 | 100% | 4062×4504 | 65.7 | 146.8 | 0 | 100% | 2412×2804 | 36.7 | 70.0 |
| Mcc2 | 0 | 100% | 17171×16830 | 575.0 | 4272.1 | 0 | 100% | 6606×6596 | 287.0 | 1592.4 |
| Struct | 0 | 100% | 8977×1597 | 12.3 | 4.8 | 0 | 100% | 6653×661 | 11.7 | 3.3 |
| Primary1 | 0 | 100% | 3335×1196 | 11.4 | 8.2 | 0 | 100% | 2085×517 | 9.2 | 4.6 |
| Primary2 | 0 | 100% | 9280×3121 | 25.4 | 54.7 | 0 | 100% | 5220×980 | 19.3 | 27.0 |
| S5378 | 0 | 100% | 3190×2890 | 15.8 | 8.7 | 0 | 100% | 1215×1661 | 11.6 | 4.5 |
| S9234 | 0 | 100% | 2601×2661 | 12.3 | 5.9 | 0 | 100% | 1066×1652 | 10.2 | 3.4 |
| S13207 | 0 | 100% | 5895×5625 | 43.0 | 37.3 | 0 | 100% | 2198×3547 | 22.5 | 18.0 |
| S15850 | 0 | 100% | 6505×5937 | 44.5 | 48.1 | 0 | 100% | 2313×3577 | 31.9 | 24.1 |
| S38417 | 0 | 100% | 11841×10557 | 81.5 | 118.9 | 0 | 100% | 3597×6224 | 72.2 | 62.3 |
| S38584 | 0 | 100% | 14979×12092 | 785.0 | 299.9 | 0 | 100% | 4558×6856 | 699.0 | 252.0 |
| Comp. | | 1 | 4.65 | 1.41 | 1.90 | | 1 | 1 | 1 | 1 |

As shown in the Table V, MARS, MGR, and MR all achieve 100% routing completion for the set of 11 benchmark circuits, but MGR has significantly better runtime efficiency than the other two works. For example, MGR obtains 6.38 times speedup over MR and about 2.16 times over MARS (note that it is hard to make a fair comparison between MARS and MGR, because MARS and MGR ran on different machines. Nevertheless, they both ran on Sun workstations. Therefore, we try our best to make a fair comparison by normalizing the runtime based on their clock rates).

*2) MCNC Benchmarks With Nonuniform Nets:* We also performed experiments on the MCNC benchmarks of nonuniform wire widths. We modify the original MCNC benchmarks of uniform wire sizes to generate a set of benchmarks of nonuniform wire sizes by using the following rules, which were proposed by [14]. The longest 10% nets are widened to twice the original width, while the next 10% are widened to 150% the original width. However, because the benchmarks S5378-S38584 are standard-cell designs, widening any pin violates the design rules for via spacing. Therefore, it is unreasonable and incorrect to test these six modified benchmarks. Since MR is a grid-based router and the routing results of MR with nonuniform wire widths violate the design rules, we do not compare MGR with MR.

Table VI gives the routability comparison of MGR with MARS. In the table, "#Total Subnets" denotes the total number of two-pin connections seen by the detailed router of MARS, since the detailed router of MARS segments long two-pin

TABLE V
COMPARISON AMONG (A) MR: MULTILEVEL GRID-BASED ROUTING WITH THE ROUTABILITY MODE [6], [31], (B) MARS: MULTILEVEL GRIDLESS GLOBAL ROUTING + FLAT GRIDLESS DETAILED ROUTING [14], AND (C) MGR. NOTE: (A) AND (C) RAN ON A 1-GHZ SUN BLADE-2000 WITH 8-GB MEMORY AND (B) RAN ON A 440-MHZ SUN ULTRA-10 WITH 384-MB MEMORY (*: FOR FAIR COMPARISONS, WE NORMALIZE THE RUNTIMES OF MARS BY THE FACTOR 440/1000)

| Circuit | (A) Results of MR | | | (B) Results of MARS | | | (C) Our Results | | |
|---|---|---|---|---|---|---|---|---|---|
| | #Failed Connections | Comp. Rates | Time (sec) | #Failed Connections | Comp. Rates | Time (sec) | #Failed Connections | Comp. Rates | Time (sec) |
| Mcc1 | 0 | 100% | 108.3 | 0 | 100% | 105.1 | 0 | 100% | 70.0 |
| Mcc2 | 0 | 100% | 3961.7 | 0 | 100% | 1916.9 | 0 | 100% | 1592.4 |
| Struct | 0 | 100% | 80.7 | 0 | 100% | 31.6 | 0 | 100% | 3.3 |
| Primary1 | 0 | 100% | 89.0 | 0 | 100% | 33.5 | 0 | 100% | 4.6 |
| Primary2 | 0 | 100% | 420.0 | 0 | 100% | 162.7 | 0 | 100% | 27.0 |
| S5378 | 0 | 100% | 6.0 | 0 | 100% | 30.0 | 0 | 100% | 4.5 |
| S9234 | 0 | 100% | 4.1 | 0 | 100% | 22.8 | 0 | 100% | 3.4 |
| S13207 | 0 | 100% | 20.8 | 0 | 100% | 85.2 | 0 | 100% | 18.0 |
| S15850 | 0 | 100% | 31.1 | 0 | 100% | 107.1 | 0 | 100% | 24.1 |
| S38417 | 0 | 100% | 70.3 | 0 | 100% | 250.9 | 0 | 100% | 62.3 |
| S38584 | 0 | 100% | 171.9 | 0 | 100% | 466.1 | 0 | 100% | 252.0 |
| Comp. | | 1 | 6.38 | | 1 | 2.16* | | 1 | 1 |

TABLE VI
COMPARISON OF (A) MARS: MULTILEVEL GRIDLESS GLOBAL ROUTING + FLAT GRIDLESS DETAILED ROUTING [14] AND (B) MGR. NOTE: (A) RAN ON A 440-MHZ SUN ULTRA-10 WITH 384-MB MEMORY AND (B) RAN ON A 1-GHZ SUN BLADE-2000 WITH 8-GB MEMORY (NOTE THAT BECAUSE THE BENCHMARK CIRCUITS S5378-S38584 VIOLATE THE DESIGN RULES OF VIA SPACING, WE DID NOT LIST THESE CASES IN THIS TABLE). (*: FOR FAIR COMPARISONS, WE NORMALIZE THE RUNTIMES OF MARS BY THE FACTOR 440/1000)

| Circuit | (A) Results of MARS | | | (B) Our Results | | |
|---|---|---|---|---|---|---|
| | #Failed Subnets (#Total Subnets) | Comp. Rates | Time (sec) | #Failed Connections | Comp. Rates | Time (sec) |
| vd_Mcc1 | 0 | 100% | 148.1 | 0 | 100% | 158.4 |
| vd_Mcc2 | 27 (99715) | 99.97% | 3388.8 | 0 | 100% | 5798.0 |
| vd_Struct | 0 | 100% | 36.3 | 0 | 100% | 7.1 |
| vd_Primary1 | 0 | 100% | 47.4 | 0 | 100% | 13.9 |
| vd_Primary2 | 0 | 100% | 296.7 | 0 | 100% | 97.1 |
| Comp. | | 99.99% | 1.15* | | 1 | 1 |

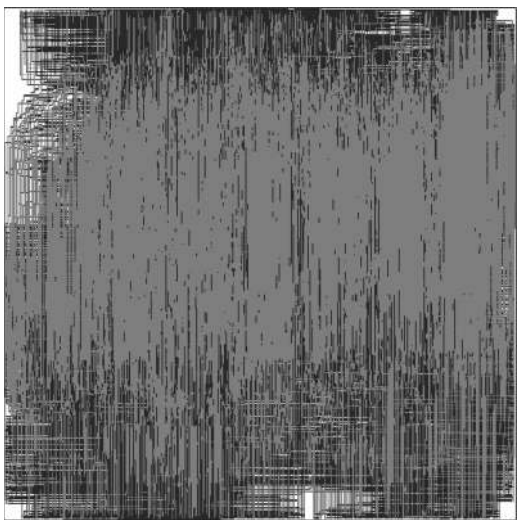

Fig. 11. Full-chip routing solution for "vd_Mcc2" obtained from MGR. The bounding box is the boundary of this benchmark circuit.

connections into short subnets. As shown in the table, MGR still achieves 100% routing completion for all of the five circuits with 1.15 times runtime speedup, while MARS completed routing for only four circuits. Note that MGR is the first router to complete the routing for this set of benchmarks of nonuniform wire sizes. In particular, we expect that the difference will be much more significant for larger and difficult designs such as vd_Mcc2. Figs. 11 and 12 show the full-chip and partial routing solutions for "vd_Mcc2" obtained from MGR, respectively.

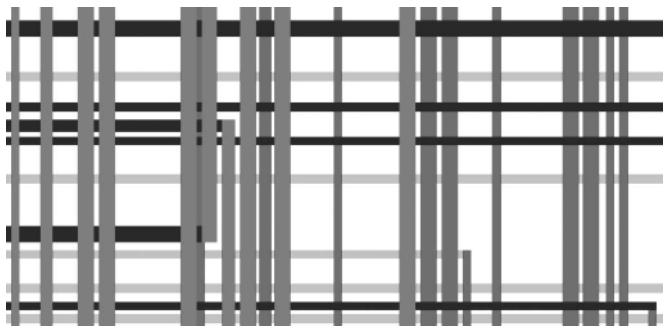

Fig. 12. Partial layout for "vd_Mcc2" obtained from MGR. We can see from the layout that the three leftmost vertical lines are of different widths.

The bounding box in Fig. 11 is the boundary of this benchmark circuit. We can see in Fig. 12 that the three leftmost vertical lines have different widths.

### C. Multilevel Routing With Faraday Benchmarks

To our best knowledge, MGR is the first academic router to route real industrial designs, the Faraday benchmarks. We compared MGR only with MR, since MR is the only academic router with source codes available to the public. As shown in Table VII, MGR achieves almost 100% routing-completion rates, while MR can only achieve about 65%–80% routing-completion rates. MR cannot achieve high routing completion rates, because it works on uniform predefined grids and assumes the design rules (wire/via width/spacing) for all routing

TABLE VII
COMPARISON OF (A) MR: MULTILEVEL GRID-BASED ROUTING WITH THE ROUTABILITY MODE [6], [31], (B) MGR. NOTE: (A) AND (B) RAN ON A 1-GHZ SUN BLADE-2000 WITH 8-GB MEMORY; (NR: NO ROUTING RESULT DUE TO OUT OF MEMORY)

| Circuit | (A) Results of MR | | | (B) Our Results | | |
|---|---|---|---|---|---|---|
| | #Failed Connections | Comp. Rates | Time (sec) | #Failed Connections | Comp. Rates | Time (sec) |
| DMA | 12856 | 64.4% | 884.0 | 16 | 99.9% | 1334.1 |
| DSP1 | 13197 | 79.2% | 1771.1 | 0 | 100% | 1551.9 |
| DSP2 | 13926 | 78.0% | 1379.9 | 0 | 100% | 1404.3 |
| RISC1 | NR | NR | NR | 0 | 100% | 3260.9 |
| RISC2 | NR | NR | NR | 0 | 100% | 3120.0 |

TABLE VIII
COMPARISON OF OUR ROUTABILITY-DRIVEN AND OPC-AWARE MGR

| Circuit | Routability-Driven MGR | | | | OPC-Aware MGR | | | |
|---|---|---|---|---|---|---|---|---|
| | #Pattern Features | #Failed Connections | Comp. Rates | Time (sec) | #Pattern Features | #Failed Connections | Comp. Rates | Time (sec) |
| Mcc1 | 40656 | 0 | 100% | 70.0 | 39772 | 0 | 100% | 101.0 |
| Mcc2 | 208132 | 0 | 100% | 1592.4 | 204010 | 0 | 100% | 2401.0 |
| Struct | 149142 | 0 | 100% | 3.3 | 128296 | 0 | 100% | 4.1 |
| Primary1 | 41110 | 0 | 100% | 4.6 | 38572 | 0 | 100% | 6.2 |
| Primary2 | 98364 | 0 | 100% | 27.0 | 86744 | 0 | 100% | 31.9 |
| S5378 | 78104 | 0 | 100% | 4.5 | 71576 | 0 | 100% | 8.9 |
| S9234 | 55982 | 0 | 100% | 3.4 | 53124 | 0 | 100% | 7.6 |
| S13207 | 154098 | 0 | 100% | 18.0 | 138624 | 0 | 100% | 28.7 |
| S15850 | 199704 | 0 | 100% | 24.1 | 173100 | 0 | 100% | 32.3 |
| S38417 | 631032 | 0 | 100% | 62.3 | 596818 | 0 | 100% | 92.0 |
| S38584 | 676250 | 0 | 100% | 252.0 | 621032 | 0 | 100% | 341.4 |
| Comp. | 1.09 | | 1 | 0.68 | 1 | | 1 | 1 |

layers to be the same. Consequently, the predefined grids are of equal size for all routing layers. However, the design rules of the Faraday benchmarks are not the same for the six routing layers. To ensure that MR satisfies the design rules, we make the pitch of its predefined grids the maximum sum of the wire/via widths and spacings. This restriction makes MR consume more routing resource in lower routing layers, resulting in inferior routing solutions.

### D. OPC-Aware Multilevel Routing

In the last experiment, we performed experiments on OPC-aware routing. For OPC-aware routing, no previous routers are available to us for comparative studies (the only work on OPC-aware routing in the literature is by Huang and Wong [22], which is a flat grid-based maze router that handles only hundreds of two-pin connections). We refer to a line as a long line if its length is five times longer than the minimum wire width. We set $L_L/W_L$ to 10/2 times of the minimum wire width. Besides, we set the effective range of OPC effect to three times of the wire pitch.

The results are listed in Table VIII. In the table, "#Pattern Features" denotes the total number of pattern features. Compared with our routibility-driven MGR, the experimental results show that our OPC-aware MGR achieves an average 9% reduction in the number of pattern features required and still maintains 100% routing completion for all circuits, with very small overheads in the runtime. The results show the effectiveness of our OPC-aware multilevel router.

## V. CONCLUSION

In this paper, we have introduced a gridless-routing model, which can obtain design-rule-correct paths and avoid redundant wires. Based on the gridless-routing model, we have proposed the first multilevel full-chip gridless detailed router. The router can handle designs with nonuniform wire/via widths and spacings and consider routability and OPC. Experimental results have shown that our approach achieves the best routing solutions in smaller running times than previous works, based on a set of commonly used MCNC benchmarks (with uniform and nonuniform wire widths) and the real industrial Faraday benchmarks (with a versatile set of design rules).

## REFERENCES

[1] S. N. Adya, S. Chaturvedi, J. A. Roy, D. Papa, and I. L. Markov, "Unification of partitioning, floorplanning and placement," in *Proc. ICCAD*, Nov. 2004, pp. 550–557.
[2] C. Albrecht, "Global routing by new approximation algorithms for multicommodity flow," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 5, pp. 622–632, May 2001.
[3] C. J. Alpert, J. H. Huang, and A. B. Kahng, "Multilevel circuit partitioning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 17, no. 8, pp. 655–667, Aug. 1998.
[4] T. Chan, J. Cong, T. Kong, and J. Shinnerl, "Multilevel optimization for large-scale circuit placement," in *Proc. ICCAD*, Nov. 2000, pp. 171–176.
[5] Y.-W. Chang, K. Zhu, and D. F. Wong, "Timing-driven routing for symmetrical-array-based FPGAs," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 5, no. 3, pp. 433–450, Jul. 2000.
[6] Y.-W. Chang and S.-P. Lin, "MR: A new framework for multilevel full-chip routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 5, pp. 793–800, May 2004.
[7] T.-C. Chen and Y.-W. Chang, "Multilevel gridless routing considering optical proximity correction," in *Proc. ASPDAC*, Jan. 2005, pp. 1160–1163.
[8] J. Cohoon and D. Richards, "Optimal two-terminal $\alpha - \beta$ wire routing," *Integr. VLSI J.*, vol. 6, no. 1, pp. 35–57, May 1988.
[9] J. Cong, J. Fang, and K. Khoo, "DUNE: A multi-layer gridless routing system with wire planning," in *Proc. ISPD*, Apr. 2000, pp. 12–18.
[10] J. Cong, S. Lim, and C. Wu, "Performance driven multilevel and multiway partitioning with retiming," in *Proc. DAC*, Jun. 2000, pp. 274–279.
[11] J. Cong, J. Fang, and K. Y. Khoo, "DUNE—A multilayer gridless routing system," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 5, pp. 633–647, May 2001.

[12] J. Cong, J. Fang, and Y. Zhang, "Multilevel approach to full-chip gridless routing," in *Proc. ICCAD*, Nov. 2001, pp. 396–403.
[13] J. Cong, M. Xie, and Y. Zhang, "An enhanced multilevel routing system," in *Proc. ICCAD*, Nov. 2002, pp. 51–58.
[14] J. Cong, J. Fang, M. Xie, and Y. Zhang, "MARS—A multilevel full-chip gridless routing system," *IEEE Trans. Comput.-Aided Design Integr. Syst.*, vol. 24, no. 3, pp. 382–394, Mar. 2005.
[15] P. Gupta, F. L. Heng, and M. Lavin, "Merits of cellwise model-based OPC," in *Proc. SPIE*, 2004, vol. 5379, pp. 182–189.
[16] M. Hayashi and S. Tsukiyama, "A hybrid hierarchical global router for multi-layer VLSIs," *IEICE Trans. Fundamentals*, vol. E78-A, no. 3, pp. 337–344, 1995.
[17] J. Heisterman and T. Lengauer, "The efficient solutions of integer programs for hierarchical global routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 10, no. 6, pp. 748–753, Jun. 1991.
[18] D. Hightower, "A solution to line routing problems on the continuous plane," in *Proc. Des. Autom. Workshop*, 1969, pp. 1–24.
[19] T.-Y. Ho, Y.-W. Chang, S.-J. Chen, and D.-T. Lee, "A fast crosstalk- and performance-driven multilevel routing system," in *Proc. ICCAD*, Nov. 2003, pp. 382–387.
[20] T.-Y. Ho, Y.-W. Chang, and S.-J. Chen, "Multilevel routing with antenna avoidance," in *Proc. ISPD*, Apr. 2004, pp. 34–40.
[21] T.-Y. Ho, Y.-W. Chang, S.-J. Chen, and D.-T. Lee, "Crosstalk- and performance-driven multilevel full-chip routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 6, pp. 869–878, Jun. 2005.
[22] L.-D. Huang and D. F. Wong, "Optical proximity correction (OPC)-friendly maze routing," in *Proc. DAC*, Jun. 2004, pp. 186–191.
[23] J. Jaja and S. A. Wu, "On routing two-terminal nets in the presence of obstacles," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 8, no. 5, pp. 563–570, May 1989.
[24] A. B. Kahng and Q. Wang, "Implementation and extensibility of an analytic placer," in *Proc. ISPD*, Apr. 2004, pp. 18–25.
[25] A. B. Kahng, S. Reda, and Q. Wang, "APlace: A general analytic placement framework," in *Proc. ISPD*, Apr. 2005, pp. 233–235.
[26] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Application in VLSI domain," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 7, no. 1, pp. 69–79, Mar. 1999.
[27] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Predictable routing," in *Proc. ICCAD*, Nov. 2000, pp. 110–114.
[28] C. Y. Lee, "An algorithm for path connection and its application," *IRE Trans. Electron. Comput.*, vol. EC-10, no. 3, pp. 346–365, Sep. 1961.
[29] H.-C. Lee, Y.-W. Chang, J.-M. Hsu, and H. Yang, "Multilevel floorplanning/placement for large-scale modules using $\mathrm{B}^*-\mathrm{trees}$," in *Proc. DAC*, Jun. 2003, pp. 812–817.
[30] Y.-L. Lin, Y.-C. Hsu, and F.-S. Tsai, "Hybrid routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 9, no. 2, pp. 151–157, Feb. 1990.
[31] S.-P. Lin and Y.-W. Chang, "A novel framework for multilevel routing considering routability and performance," in *Proc. ICCAD*, Nov. 2002, pp. 44–50.
[32] C. Liu, H.-P. Tseng, and C. Sechen, "Chip-level area routing," in *Proc. ISPD*, Apr. 1998, pp. 197–204.
[33] R. Eric Lunow, "A channelless, multilayer router," in *Proc. DAC*, Jun. 1988, pp. 667–671.
[34] M. Marek-Sadowska, "Global router for gate array," in *Proc. ICCD*, Oct. 1984, pp. 332–337.
[35] M. Marek-Sadowska, "Router planner for custom chip design," in *Proc. ICCAD*, Nov. 1986, pp. 246–249.
[36] A. Margarino, A. Romano, A. De Gloria, F. Curatelli, and P. Antognetti, "A tile-expansion router," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. CAD-6, no. 4, pp. 507–517, Jul. 1987.
[37] G. Meixner and U. Lauther, "A new global router based on a flow model and linear assignment," in *Proc. ICCAD*, Nov. 1990, pp. 44–47.
[38] T. Ohtsuki, "Gridless routers-new wire routing algorithms based on computational geometry," in *Proc. Int. Conf. Circuits and Syst.*, May 1985, pp. 802–809.
[39] J. K. Ousterhout, "Corner stitching: A data-structure technique for VLSI layout tools," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. CAD-3, no. 1, pp. 87–100, Jan. 1984.
[40] M. Sato, J. Sakanaka, and T. Ohtsuki, "A fast line-search method based on a tile plane," in *Proc. IEEE Int. Symp. Circuits and Syst.*, May 1987, pp. 588–591.
[41] W. Schiele, T. Kruger, K. Just, and F. Kirsch, "A gridless router for industrial design rules," in *Proc. DAC*, Jun. 1990, pp. 626–631.
[42] J. Soukup, "Fast maze router," in *Proc. DAC*, Jun. 1978, pp. 100–102.
[43] C. Tsai, S. Chen, and W. Feng, "An H-V alternating router," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 11, no. 8, pp. 976–991, Aug. 1992.
[44] D. Wang and E. Kuh, "A new timing-driven multilayer MCM/IC routing algorithm," in *Proc. Multi-chip Module Conf.*, Feb. 1997, pp. 89–94.
[45] Y. F. Wu, P. Widmayer, M. D. F. Schlag, and C. K. Wong, "Rectilinear shortest paths and minimum spanning trees in the presence of rectilinear obstacles," *IEEE Trans. Comput.*, vol. C-36, no. 3, pp. 321–331, Mar. 1987.
[46] Y.-R. Wu, M.-C. Tsai, and T.-C. Wang, "Maze routing with OPC consideration," in *Proc. ASPDAC*, Jan. 2005, pp. 198–203.
[47] Z. Xing and R. Kaog, "Shortest path search using tiles and piecewise linear cost propagation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 2, pp. 145–158, Feb. 2002.
[48] S. Q. Zheng, J. S. Lim, and S. S. Iyengar, "Finding obstacle-avoiding shortest paths using implicit connection graphs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 1, pp. 103–110, Jan. 1996.

**Tai-Chen Chen** (S'02) received the B.S. and M.S. degrees in computer and information science from National Chiao Tung University, Hsinchu, Taiwan, in 1999 and 2001, respectively. He is currently working toward the Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan.

His current research interests include computer-aided design and gridless routing for nanometer electrical effects.

Mr. Chen is the recipient of the Best M.S. Thesis Award from the National Science Council of Taiwan in 2002.

**Yao-Wen Chang** (S'94–M'96) received the B.S. degree from the National Taiwan University, Taipei, Taiwan, in 1988 and the M.S. and Ph.D. degrees from the University of Texas at Austin in 1993 and 1996, respectively, all in computer science.

He is a Professor in the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University. He is currently also a Visiting Professor at Waseda University, Kitakyushu, Japan. He was with the IBM T.J. Watson Research Center, Yorktown Heights, NY, in the summer of 1994. From 1996 to 2001, he was on the faculty of the National Chiao Tung University, Hsinchu, Taiwan. He has coauthored more than 110 ACM/IEEE conference/journal papers and one book on routing. His current research interests are in VLSI physical design, design for manufacturing, and FPGA. He has been working closely with industry on projects in these areas. He is an Editor of the *Journal of Computer and Information Science*.

Dr. Chang is currently the Chair of the Design Automation and Test (DAT) Consortium of the Ministry of Education, Taiwan, a member of the Board of Governors of Taiwan IC Design Society, and a member of the IEEE Circuits and Systems Society, ACM, and ACM/SIGDA. He currently serves on the ACM/SIGDA Physical Design Technical Committee and the technical program committees of important conferences on VLSI design automation, including ASP-DAC (Topic Chair), DAC, DATE, FPT, GLSVLSI, ICCAD, ICCD, ISPD, SOCC (Topic Chair), and VLSI-DAT (Topic Chair). He was a recipient of the 2006 ACM ISPD Placement Contest Award, Best Paper Award at ICCD-1995, and eight Best Paper Nominations from DAC-2007, ISPD-2007, DAC-2005, 2004 ACM TODAES, ASP-DAC-2003, ICCAD-2002, ICCD-2001, and DAC-2000. He was also a recipient of many awards for research performance, such as the 2005 and 2006 First-Class Principal Investigator Awards and the 2004 Mr. Wu Ta You Memorial Award from National Science Council of Taiwan, the 2004 MXIC Young Chair Professorship from the MXIC Corporation, and for excellent teaching from the National Taiwan University and National Chiao Tung University.