Full-Chip Routing Considering Double-Via Insertion

Huang-Yu Chen, *Student Member, IEEE*, Mei-Fang Chiang, *Student Member, IEEE*, Yao-Wen Chang, *Member, IEEE*, Lumdo Chen, and Brian Han

***Abstract*—As the technology node advances into the nanometer era, via-open defects are one of the dominant failures due to the copper cladding process. To improve via yield and reliability, redundant-via insertion is a highly recommended technique proposed by foundries. Traditionally, double-via insertion is performed at the postlayout stage. The increasing design complexity, however, leaves very limited space for postlayout optimization. It is thus desirable to consider the double-via insertion at both the routing and postrouting stages. In this paper, we present a new full-chip gridless routing system considering double-via insertion for yield enhancement. To fully consider double vias, the router applies a novel two-pass, bottom-up routability-driven routing framework and features a new redundant-via aware detailed maze routing algorithm (which could be applied to both gridless and grid-based routing). We also propose a graph-matching based post-layout double-via insertion algorithm to achieve a higher insertion rate. In particular, the algorithm is optimal for grid-based routing with up to three routing layers and the stacked-via structure. Experiments show that our methods significantly improve the via count, number of dead vias, double-via insertion rates, and running times.**

***Index Terms*—Design for manufacturability, detailed routing, double via, global routing, gridless routing, physical design, redundant via.**

I. Introduction

As integrated circuit process geometries shrink to 65 nm and below, yield and reliability become first-order cost metrics. Via-open defects are one of the important failures. A via may fail due to various reasons such as random defects, electromigration, cut misalignment, and/or thermal-stress-induced voiding effects. Via failures significantly reduce the manufacturing yield and chip performance.

Manuscript received February 24, 2007; revised July 21, 2007. This work was supported in part by the National Science Council of Taiwan under Grant NSC 96-2752-E-002-008-PAE, Grant NSC 96-2628-E-002-248-MY3, Grant NSC 96-2628-E-002-249-MY3, and Grant NSC 96-2221-E-002-245. This paper was presented in part at the 43rd Design Automation Conference, July 24–28, 2006, San Francisco, CA. This paper was recommended by Associate Editor L. Scheffer.

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

M.-F. Chiang is with the Graduate School of Information, Production and Systems, Waseda University, Kitakyushu 808-0135, Japan (e-mail: annika@ruri.waseda.jp).

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

L. Chen and B. Han are with United Microelectronics Corporation, Hsinchu 300, Taiwan, R.O.C. (e-mail: lumdo_chen@umc.com; soe_han@yahoo.com).

Digital Object Identifier 10.1109/TCAD.2008.917597

A. Redundant-Via Insertion

To improve via yield and reliability, redundant-via insertion is a highly recommended technique proposed by foundries. If one via fails, a redundant via can serve as a fault-tolerant substitute for the failing one. As reported in [24], double vias lead to 10×–100× smaller failure rates than single vias. Existing approaches are often for postlayout optimization by replacing a single via with a double-via structure as long as it does not create any design-rule violations [3], [18]. The increasing design complexity, however, leaves very limited space for postlayout optimization. It has been reported that inserting redundant vias during routing can improve the double-via insertion rates by 15%–25% than the postlayout optimization, at the cost of routability degradation. As a result, major foundry, such as the Taiwan Semiconductor Manufacturing Company, has recommended performing double-via insertion during routing and postlayout optimization as a defect-tolerant process for 90- and 65-nm manufacturing processes [22]. Therefore, it is desired to consider the double-via insertion at both the routing and postrouting stages for better tradeoff in routability and double-via insertion rates.

Recently, Xu *et al.* [26] proposed pioneering work to consider double-via insertion during maze routing. By assigning double-via costs to the routing graph, they formulated the problem as a multiobjective maze routing problem and applied Lagrangian relaxation to solve it. However, they only consider the redundant via at the detailed routing stage, and the high time complexity of Lagrangian relaxation limits the feasible problem size to be within hundreds of nets. Yao *et al.* [27] developed a grid-based router, which features via-minimization global routing, followed by double-via aware detailed routing; the work claimed that the postlayout double-via insertion (PDVI) problem can be solved by a maximum bipartite matching formulation, which was recently shown to be incorrect for some cases in [18]. Lee and Wang [18] instead formulated the double-via insertion problem as a maximum independent set (MIS) problem. Since the MIS problem for a general graph is nondeterministic polynomial time (NP)-complete [15], they resorted to heuristics to handle the problem.

B. Routing Framework Evolution

The continuously increasing design complexity imposes severe challenges for modern routers. Nowadays, a modern chip may contain several billion transistors and has over one million nets. Consequently, the routing frameworks are evolving from the *flat* framework to the *hierarchical* and *multilevel* frameworks. Many routing algorithms adopt a two-stage flat

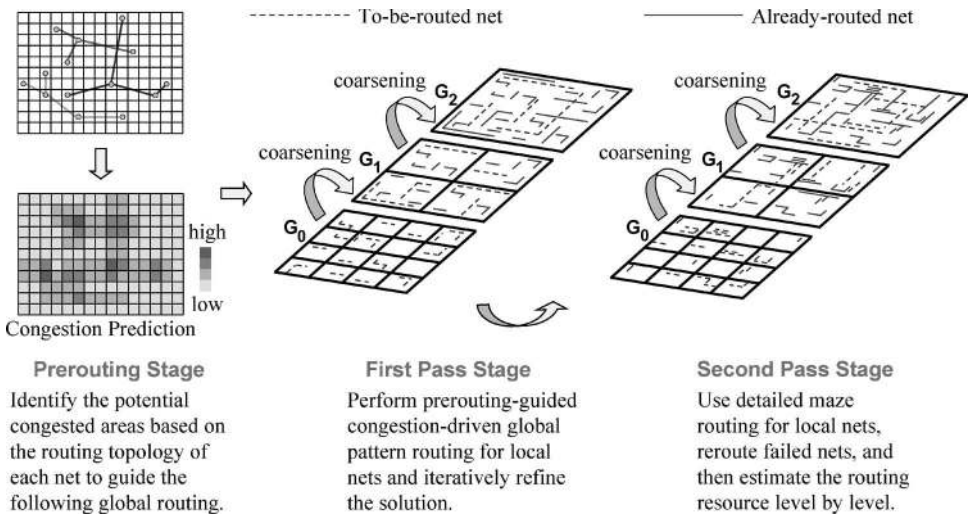

Fig. 1. New two-pass bottom-up routing framework.

framework of global routing, followed by detailed routing [2], [17]. Global routing first partitions the routing region into tiles and then decides tile-to-tile paths for all nets, whereas detailed routing determines exact tracks and vias for nets within each tile.

However, the flat framework does not scale well as the design size increases. To cope with the scalability problem, hierarchical and multilevel frameworks are proposed. The hierarchical framework uses the divide-and-conquer approach to handle smaller subproblems independently [12], [19], [21]. Although the hierarchical approach can scale to larger designs, it has a drawback of lacking interactions among routing subregions and thus limits the solution quality. To remedy the deficiencies, researchers have proposed various multilevel frameworks to handle large-scale routing problems. The traditional Λ-shaped multilevel routing framework consists of bottom-up *coarsening*, followed by top-down *uncoarsening* (e.g., MARS [10], [11], MR [6], CMR [13], and MGR [8]), whereas the V-shaped one consists of top-down *uncoarsening*, followed by bottom-up *coarsening* (e.g., VMGR [9]). The coarsening stage is a bottom-up approach that iteratively groups a set of circuit components (e.g., routing tiles) based on a predefined cost metric. In contrast, the uncoarsening stage iteratively ungroups a set of previously clustered circuit components in a top-down manner. The multilevel frameworks demonstrate the superior capability of handling large-scale routing problems and the versatility of tackling modern nanometer electrical effects, such as crosstalk [13] and optical proximity correction [8]. It is also observed that the Λ-shaped multilevel framework can handle local circuit effects (such as routability, congestion, and via minimization) better since it works in a bottom-up manner and deals with local routing regions first (i.e., route shorter local nets and then longer global nets) [6], [16]. In contrast, the V-shaped multilevel framework is more suitable for handling global electrical effects (such as crosstalk and critical-path delay) since it works in a top-down manner and copes with global routing regions first [9].

C. Our Contributions

To maximize the redundant-via insertion rate, we consider redundant-via insertion during routing as well as postlayout optimization. In this paper, we present a new full-chip gridless routing system called Two-pass Bottom-up gridless Router (TBR), considering redundant-via insertion for yield enhancement. To fully consider redundant vias, the router is based on a novel two-pass bottom-up routability-driven routing framework. Different from the previous routing frameworks, TBR adopts a three-stage technique of a *prerouting* stage, followed by a bottom-up *global* routing stage and then followed by a bottom-up *detailed* routing stage. Fig. 1 illustrates an the new framework. The motivation for the two-pass bottom-up approach lies in the observation that it is more effective to route shorter local nets first for routability and via optimization. Although the Λ-shaped multilevel framework also performs bottom-up coarsening first, it refines the solution top-down during the following uncoarsening stage. By using the two-pass bottom-up approach, we can take full advantage of processing local nets first at the two routing passes for routability and via optimization.

For congestion minimization, TBR starts with a prerouting stage, which estimates the potential congested areas based on the routing topology of each net. Guided by this estimation, the following routing would favor the paths that minimize congestions. Then, the first bottom-up global routing pass is performed; it starts from coarsening the finest level to the coarsest level. At each level, the L- or Z-shaped pattern routing [16] is used for congestion-driven global routing. When an initial global routing solution is generated, net-ordering dependence is explored, and iterative refinement is performed for congestion optimization. It will be clear that the L- and Z-shaped global pattern routing leads to the via count reduction in the following detailed routing stage. The second bottom-up routing pass is for detailed routing. As in the first pass, it proceeds from coarsening the finest level to the coarsest level. At each level, redundant-via aware detailed maze routing (could be gridless or grid-based routing) is performed, and rip-up/reroute procedures are applied for failed nets. By performing the redundant-via aware detailed routing, we can minimize the number of vias that do not have any feasible redundant-via candidates and reserve more adjacent space for each via to facilitate the PDVI and thus increase the final redundant-via insertion rate.

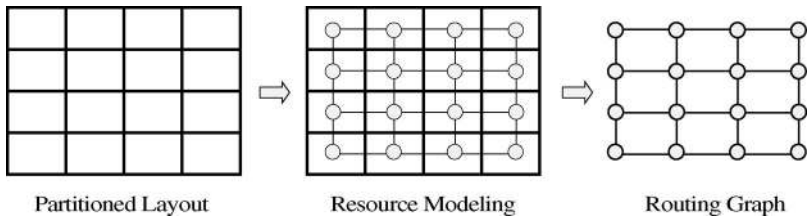

Fig. 2. Routing graph.

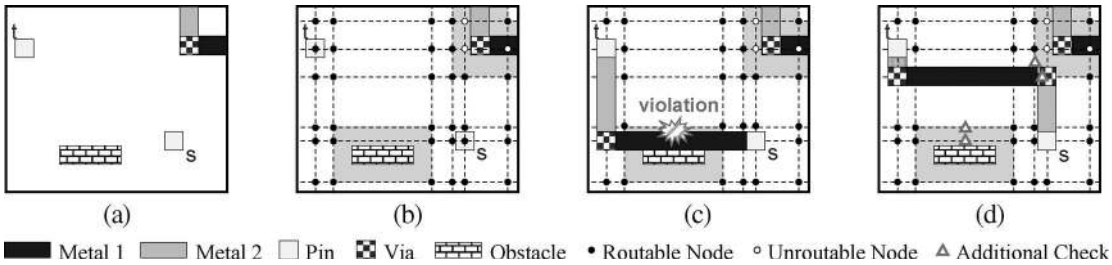

Fig. 3. Gridless detailed routing model. (a) Detailed routing instance. (b) Implicit connection graph. (c) Infeasible detailed routing path. (d) Design-rule-correct detailed routing path.

We also propose a graph-matching based post-layout double-via insertion algorithm to achieve a higher insertion rate. In particular, the algorithm is optimal for grid-based routing with up to three routing layers and the stacked-via structure. With the optimal algorithm for the restricted problem, we then extend it to handle the general case of any routing layer and via structure.

Experimental results show that our routing system reduces the via count by 1.20× compared with the state-of-the-art gridless router [8]–[10], and our redundant-via aware detailed router can effectively obtain fewer dead vias by 1.44×. Compared with the state-of-the-art double-via insertion algorithm [18] under the stacked-via formulation, our post-layout double-via insertion algorithm can achieve 98.6% double-via insertion rate with at least 70.8× runtime speedup.

The rest of this paper is organized as follows: Section II describes the routing model and the postlayout redundant-via insertion problem. Section III presents our novel two-pass bottom-up routing framework considering redundant-via insertion. Section IV presents our PDVI algorithm. Experimental results are reported in Section V, and conclusions are given in Section VI.

II. Preliminaries

In this section, we give the routing models, the two-pass bottom-up routing framework, and the formulation for the PDVI problem.

A. Global Routing Model

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

Before we can apply the graph-search technique to routing, we first need to model the routing resource as a *routing graph* whose topology can represent the chip structure. Fig. 2 illustrates the graph modeling. For the modeling, we first partition a chip into an array of rectangular *global cells (GCs)*, each of which may accommodate tens of routing tracks in each dimension. A node in the routing graph represents a GC in the chip, whereas an edge denotes the boundary between two adjacent GCs. Each edge is assigned a capacity according to the physical area or the size of a GC. A global router finds GC-to-GC paths for all nets to guide the detailed router. The goal of global routing is to route as many nets as possible without violating any capacity constraint of each edge and to meet any other specified optimization constraints.

B. Detailed Gridless Routing Model

The goal of detailed routing is to find a design-rule-correct path for each connection while meeting every specified constraint. Our gridless detailed routing applies a graph-search technique based on an *implicit connection graph* used in [10], with a modification to guarantee the correctness of the searched path. Fig. 3(a) gives a detailed routing instance with source s and target t, and Fig. 3(b) shows the corresponding implicit connection graph constructed based on s, t, and the *obstacle zones*. An obstacle zone is a minimum expansion of an existing obstacle (e.g., an already-routed wire) such that any new routing wire lying on the boundaries of this obstacle zone would not violate any design rule. A node in the implicit connection graph is an *unroutable node* if it is inside an obstacle zone; it is a *routable node* otherwise. The black and white circles in Fig. 3(b) represent the routable and unroutable nodes, respectively.

It should be noted that, if there exists a path formed by two successive routable nodes belonging to the same obstacle zone, it might be considered a feasible route in [10]. As shown in Fig. 3(c), nevertheless, a path might illegally cut across one obstacle zone through two routable nodes lying on the boundaries of this obstacle zone. To guarantee the correctness of the searched path, Chen *et al.* [8], [9] added horizontal and vertical middle lines for each obstacle zone. However, this approach would significantly increase the search space and,

thus, the running time. To remedy this deficiency, we search the path by an additional check to see if the midpoint of the two routable nodes belonging to the same obstacle zone is an interior point of this obstacle zone. Fig. 3(d) shows the triangular points representing these additional checks. Note that each check operation takes only constant time. In this way, a design-rule-correct detailed routing path can be efficiently guaranteed.

C. Two-Pass Bottom-Up Routing Framework Model

As illustrated in Fig. 1, G_k corresponds to the routing graph of level k. The first bottom-up routing pass is the global routing stage, which starts from coarsening the finest level (level 0) to the coarsest level. At each level k, our global router finds routing paths for the local connections (those connections that entirely sit inside GC_{k+1}, where GC_k is the GC of level k). After global routing of level k is performed, we merge four GC_ks into a larger GC_{k+1} and, at the same time, perform resource estimation for use at the next level $k+1$. Coarsening continues until the number of GCs at a level is below a threshold.

The second bottom-up routing pass is the detailed routing stage. As the first pass, it processes from coarsening the finest level to the coarsest level. At each level, a detailed maze routing is performed, and rip-up/reroute procedures are applied for failed nets. The process continues until we reach the coarsest level when the final routing solution is obtained.

D. PDVI

After the detailed routing, we have a resulting layout with various types of vias. For a via, a *redundant-via candidate* is a candidate position where a redundant via can be inserted for this via without violating any design rule. Note that a redundant via is typically horizontally or vertically inserted at a *fixed* distance from a via for practical applications; thus, we can treat a via and its redundant via as a pair. Therefore, we refer to a redundant-via candidate as one of the four positions with a fixed distance (specified by the design rules) from a via in the left, right, top, and bottom directions, same as most existing works [3], [18]. A via is an *alive via* if it has at least one redundant-via candidate for insertion; otherwise, it is a *dead via*. A *critical via* refers to an alive via with exactly one redundant-via candidate. A via is either a *single (conventional) via* or a *stacked via*. A stacked via is the via consisting of at least two vertically stacked single vias.

We treat both the stacked via and the single via as one unit via since, from the connection viewpoint, if any one single via contained in a stacked via is not paired with a redundant via, this stacked via is still not protected. The formulation of the post-layout redundant-via insertion problem is defined as follows.

- Problem PDVI: Given a postrouting layout, pair each alive via with a redundant via as many as possible such that no design rule is violated after the double-via insertion.

III. Redundant-Via Aware Routing

For via yield enhancement during routing, we shall try to: 1) minimize the via count to reduce the failure probability and 2) plan the double-via positions for each via to ensure that a double via can be inserted wherever needed in the later PDVI process.

To deal with the simultaneous optimization, we propose a novel two-pass bottom-up routing framework that adopts a three-stage technique of a prerouting stage, followed by a bottom-up *global* routing stage and a bottom-up *detailed* routing stage (see Fig. 1). The prerouting stage identifies the potentially congested areas to guide the following routing for congestion optimization. The two bottom-up routing passes tend to route shorter nets first level by level, which directly contributes to the routability enhancements and via-count minimization. Based on this routability- and via-aware framework, we develop a two-pass bottom-up full-chip gridless routing system named TBR. Specifically, the TBR consists of the following: 1) a congestion-driven prerouting stage; 2) a via-minimization global routing stage; and 3) a redundant-via planning detailed routing stage. We detail the three stages in the following sections.

A. Congestion-Driven Prerouting Stage

To improve routability, the works [6], [8], and [9] integrated global routing, detailed routing, and resource estimation together at each level of the framework, leading to more accurate routing resource estimation. However, this approach might confine the optimization freedom since global routing and detailed routing are intertwined with each other.

In order to consider more objectives for congestion minimization, the TBR features a prerouting stage that identifies the potentially congested areas based on the routing topology of each net. With the prerouting, the TBR can separately perform global routing and detailed routing and leave more flexibility in dealing with the redundant-via-related objectives.

Given a netlist, we first construct a minimum spanning tree (MST) for each net and then decompose each net into two-pin connections, with each connection corresponding to an edge of the MST. The MST topology leads to the minimum total wire length; thus, congestion is often easier to be controlled than other topologies. The TBR then preestimates the congestion in the routing graph for all two-pin connections using the *probabilistic congestion model*, which has recently been successfully applied to placement [4], floor planning [14], and routing [20], [25], and is generally believed to have the ability to alleviate the net-ordering problem in sequential routing. The TBR preevaluates the congestion as the average number of global one-bend and two-bend routes that might pass through the boundary of adjacent GCs. For a two-pin connection c, we first explore all possible one- and two-bend global routes from its source s to its target t, which are denoted by set P_c. All routes in P_c are the candidates of global routing solutions for c. For a boundary b_i between two GCs, let $N_c(i) = \{r \in P_c | r \text{ is the route passing through } b_i\}$; then, the estimated congestion of b_i with respect to c is equal to $|N_c(i)|/|P_c|$. For example, as shown in Fig. 4(a), connection c has five possible one- and two-bend routes from source s to target t. Fig. 4(b) gives the number of routes passing through each GC boundary b_i, $|N_c(i)|$, and Fig. 4(c) shows the congestion estimation of c in the routing graph.

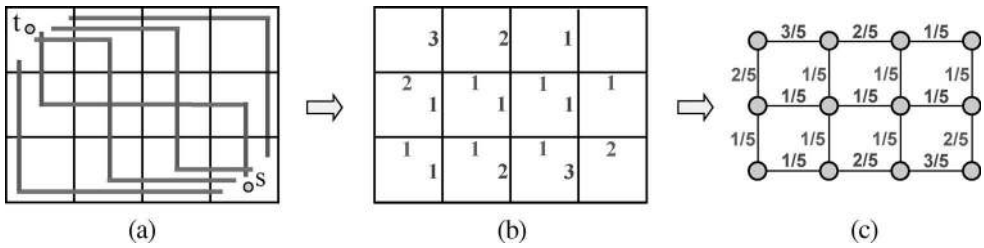

Fig. 4. Probabilistic congestion estimation. (a) Two one-bend and three two-bend routes from s to t. (b) Number of routes through each boundary. (c) Preestimation congestion in the routing graph.

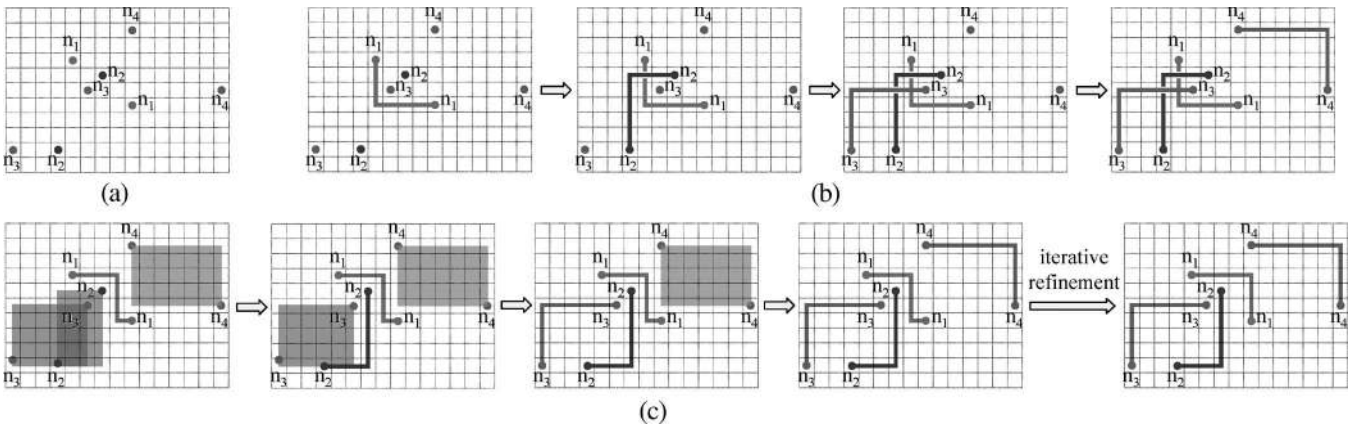

Fig. 5. Effect of congestion estimation. (a) Global-routing instance. (b) Routing result generated by a traditional net-ordered global router. (c) Better routing result generated by the prerouting-guided congestion-driven global router, followed by iterative refinement.

B. Via-Minimization Global Routing Stage

The first bottom-up global-routing pass is a coarsening process starting from the finest level to the coarsest level. Our global routing is based on the approach used for pattern routing [16]. Let the routing graph of level 0 be $G_0 = (V_0, E_0)$ and the global routing result for a local connection c be $R_c = \{e \in E_0 | e \text{ is the edge chosen for routing}\}$. For the congestion control, we define the cost function of the global routing result R_c as follows:

$$\Psi_{R_c} = \alpha \max_{e \in R_c}(c_e) + \frac{\beta}{|R_c|} \sum_{e \in R_c} c_e \tag{1}$$

where α and β are user-defined parameters, and c_e denotes the congestion of edge e, which is defined by

$$c_e = d_e / p_e \tag{2}$$

where d_e and p_e are the density and capacity associated with e, respectively. By dynamic density, pattern routing uses an L-shaped (one-bend) or Z-shaped (two-bend) route to make the connection, which gives the shortest path length between two points. Therefore, the wire length is minimum; thus, we do not include it in the cost function at this stage. This cost function can guide our global router to select a path with smaller maximum and average congestion. Note that density d_e in (2) comes from both the predicted congestion obtained at the prerouting stage and real routing. Its value is dynamically updated as the routing proceeds.

Fig. 5 illustrates the effects of the prerouting-guided global routing. Fig. 5(a) gives a global-routing instance with four nets n_1, n_2, n_3, and n_4. A traditional bottom-up net-ordered router would route the nets in the order n_1, n_2, n_3, and n_4 based on their net lengths in each level, from short to long, as shown in Fig. 5(b). With the prerouting-guided congestion-driven global routing and iterative refinement, in contrast, we can proceed with the routing, as shown in Fig. 5(c), and obtain a better solution.

C. Redundant-Via Aware Detailed Routing Stage

Similar to the global-routing stage, the second bottom-up detail-routing pass is a coarsening process starting from the finest level to the coarsest level. In addition to the routability consideration, we shall also maximize the possibility for postlayout redundant-via insertion in this stage. To do so, Xu *et al.* [26] considered via minimization and redundant-via planning during detailed routing. They assigned redundant-via costs to the edges of the detailed routing graph to estimate the number of *dead vias* induced by the route and applied Lagrangian relaxation to solve the problem. However, their cost assignment is not suitable for gridless routing, because there exists no uniform grid for gridless routing, and the high time complexity of Lagrangian relaxation limits their applications to only hundreds of nets.

To simultaneously consider redundant-via planning and via minimization, we define the following cost function for a net n to guide the maze routing:

$$\Phi_n = \gamma \nu(n) + \delta \rho(n) \tag{3}$$

where γ and δ are user-defined parameters, $\nu(n)$ is the number of vias in n, and $\rho(n)$ is a redundant-via related penalty function for n.

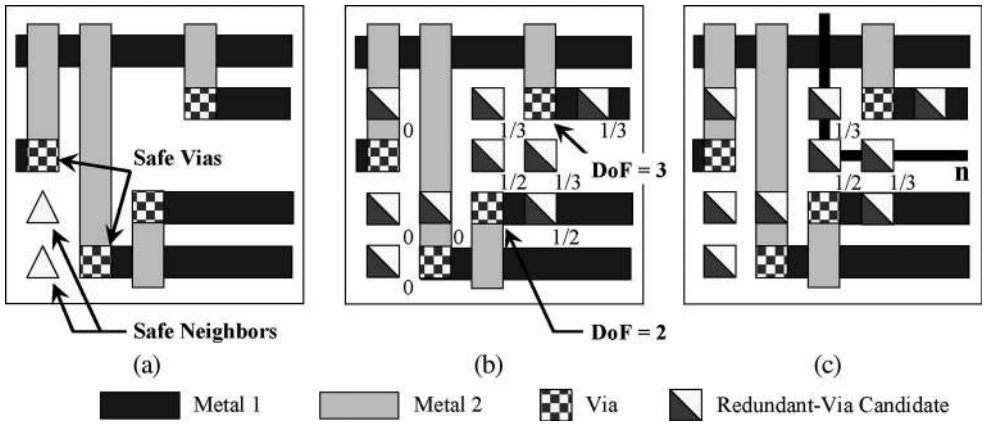

Fig. 6. Redundant-via related penalty assignment. (a) Safe neighbors and safe vias. (b) Cost assignments. (c) Penalty function of net n equals 7/6.

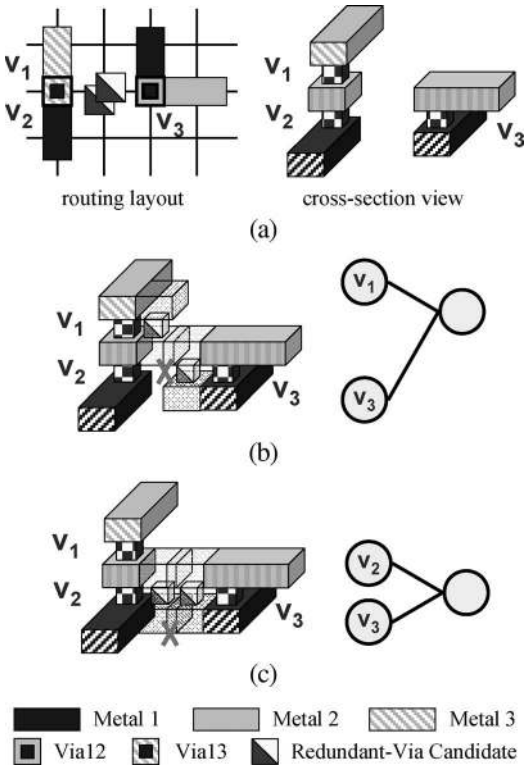

Fig. 7. Counterexample for maximum bipartite matching formulation. (a) Routing layout consists of three single vias v_1, v_2, and v_3. (b) Vertical design-rule conflict exists between the redundant-via candidates of v_1 and v_3. (c) Horizontal design-rule conflict exists between the redundant-via candidates of v_2 and v_3.

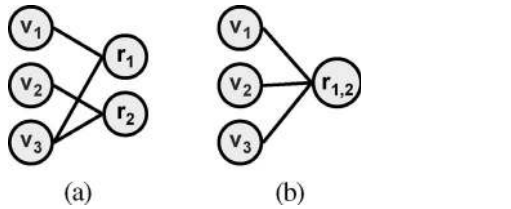

Fig. 8. Some possible bipartite graph formulations for Fig. 7(a). (a) Infeasible bipartite graph formulation. (b) Feasible bipartite graph formulation but cannot achieve the optimal solution.

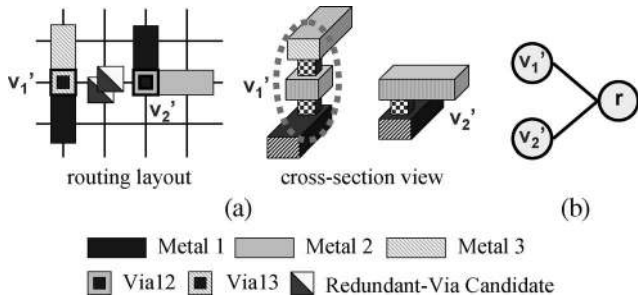

Fig. 9. (a) Stacked-via structure. (b) Our bipartite graph formulation.

In this stage, we perform the modified Lee's detailed maze routing algorithm by selecting the paths with the minimum cost of (3) among all the shortest paths. We explain the determination of $\rho(n)$. We call a redundant-via candidate a *safe neighbor* if it is always available (e.g., it has a special position such that it never vanishes due to other nets passing through it) and is not shared with any other via. A via with at least one safe neighbor is called a *safe via* [see Fig. 6(a)]. Since a safe via can always be protected by its safe neighbor, we set the cost to zero in all its redundant-via candidates. The *degree of freedom* of a via v, which is denoted by DoF_v, is the total number of redundant-via candidates of v, as defined in [26]. If via v is not a safe via, we assign the cost $1/\mathrm{DoF}_v$ to its redundant-via candidates. The rational is that, if via v becomes a dead via due to net n (i.e., n passes all the redundant-via candidates of v), the cost for routing n would be increased by $\mathrm{DoF}_v \times 1/\mathrm{DoF}_v = 1$, which is exactly equal to the number of the induced dead vias. Note that a redundant-via candidate r_c may be shared by more than one via. In this case, we set the cost of r_c as $\{\max\{1/\mathrm{DoF}_{v_i}\}|v_i$ is the via that shares $r_c\}$. Fig. 6(b) shows the redundant-via candidates and their cost assignments. Finally, the penalty function $\rho(n)$ of net n is the summation of the costs of the redundant-via candidates that are passed through by n. The $\rho(n)$ of route n in Fig. 6(c) is calculated as $1/3 + 1/2 + 1/3 = 7/6$.

IV. PDVI

Yao *et al.* [27] claimed that the PDVI problem can be straightforwardly solved by a maximum bipartite matching formulation, which was recently shown to be incorrect in [18]. Fig. 7(a) shows a counterexample consisting of three single vias v_1, v_2, and v_3. Note that v_1 and v_2 belong to the same net, which is different from the net of v_3. We can see from Fig. 7(b) that

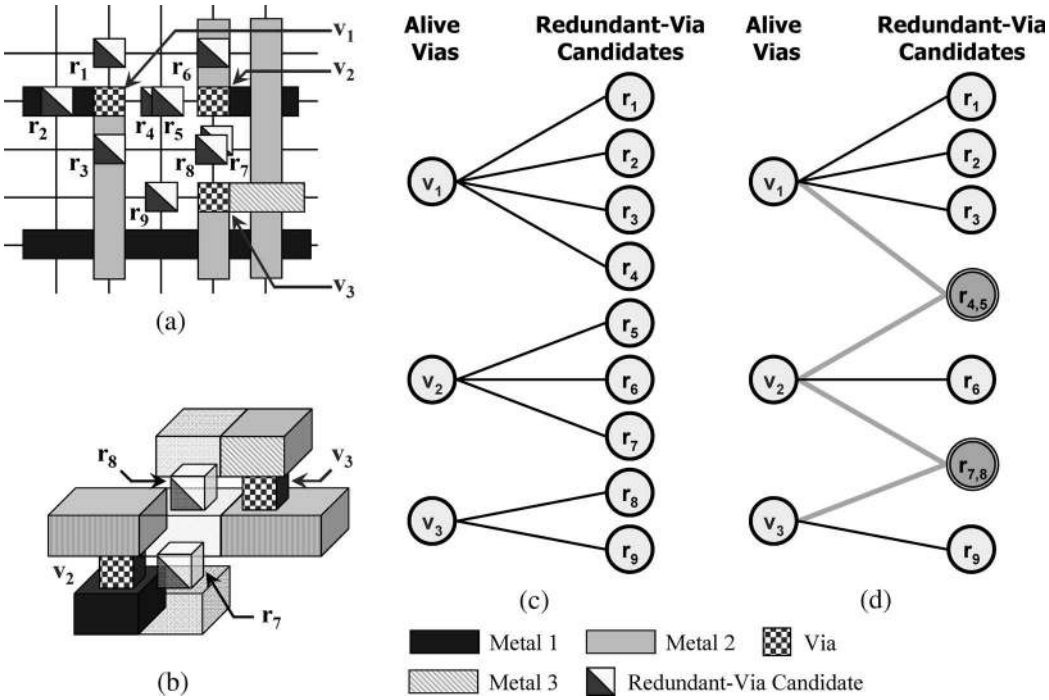

Fig. 10. Bipartite graph construction. (a) A grid-based routing layout. (b) Cross-sectional view for the vertical design-rule conflict between r_7 and r_8. (c) Initial bipartite graph without considering conflicts. (d) Final bipartite graph after merging (r_4, r_5) and (r_7, r_8).

there is a vertical design-rule conflict between the redundant-via candidates of v_1 and v_3. Therefore, vias v_1 and v_3 cannot simultaneously be paired with a redundant via, or these two nets will be short together. Thus, in the bipartite graph, v_1 and v_3 have to connect to the same node to model the constraint. Similarly, vias v_2 and v_3 cannot simultaneously be paired with a redundant via because a horizontal design-rule conflict exists between their redundant-via candidates, so they also connect to one node, as shown in Fig. 7(c).

To get the whole bipartite graph, we can simply combine the bipartite graphs together, as shown in Fig. 8(a) and (b). However, a maximum bipartite matching solution of Fig. 8(a) may lead to an infeasible solution that v_1 and v_3 are simultaneously paired with a redundant via, which is a contradiction to Fig. 7(b). On the other hand, although all maximum bipartite matching solutions of the formulation in Fig. 8(b) are feasible, these solutions cannot obtain the optimal solution in which v_1 and v_2 can be simultaneously paired. As a result, Lee and Wang [18] instead formulated the double-via insertion problem as a MIS problem. Since the MIS problem is NP-complete [15], they resorted to heuristics to handle the problem.

For the PDVI (Post-layout Double-Via Insertion) problem, we develop a polynomial-time double-via insertion algorithm based on a bipartite graph matching formulation. In particular, this graph matching based algorithm is optimal for grid-based routing with up to three routing layers (*i.e.*, two via layers) and the stacked-via structure [*i.e.*, two or more vias vertically stacked are treated as one stacked via; see v_1' in Fig. 9(a)]. With the algorithm for the restricted problem, we then extend it to handle the general case of any number of routing layers and via structure. With our method, the counterexample shown in [18] can be exactly formulated, as shown in Fig. 9(b). The maximum bipartite matching solution will be either stacked via v_1' is paired with a stacked redundant via or single via v_2' is paired with a redundant via.

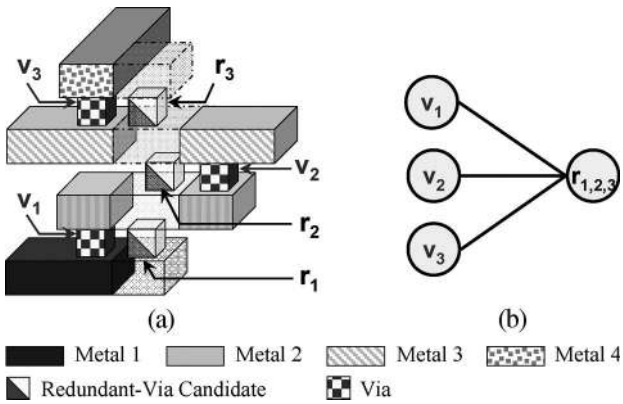

Fig. 11. (a) Four-layer routing layout. (b) The bipartite graph cannot achieve the optimal solution.

A. Optimal Algorithm for Up to Three Routing Layers

Our bipartite graph construction is given as follows: Given a routing layout with up to three layers, we first construct an undirected bipartite graph $G(V, E)$, with two disjoint vertex partitions V_A and V_C such that $V = V_A \cup V_C$. Here, V_A is the set of alive vias, and V_C is the set of redundant-via candidates. For $v_a \in V_A$ and $r_c \in V_C$, an edge $(v_a, r_c) \in E$ exists if r_c is a redundant-via candidate of v_a. Next, we merge two nodes $r_i, r_j \in V_C$ into one node $r_{i,j}$ if there is a vertical or horizontal design-rule conflict between the redundant-via candidates r_i and r_j (i.e., r_i and r_j cannot be simultaneously inserted). For example, Fig. 10(a) gives a grid-based routing layout, where a horizontal conflict exists between r_4 and r_5 and a vertical

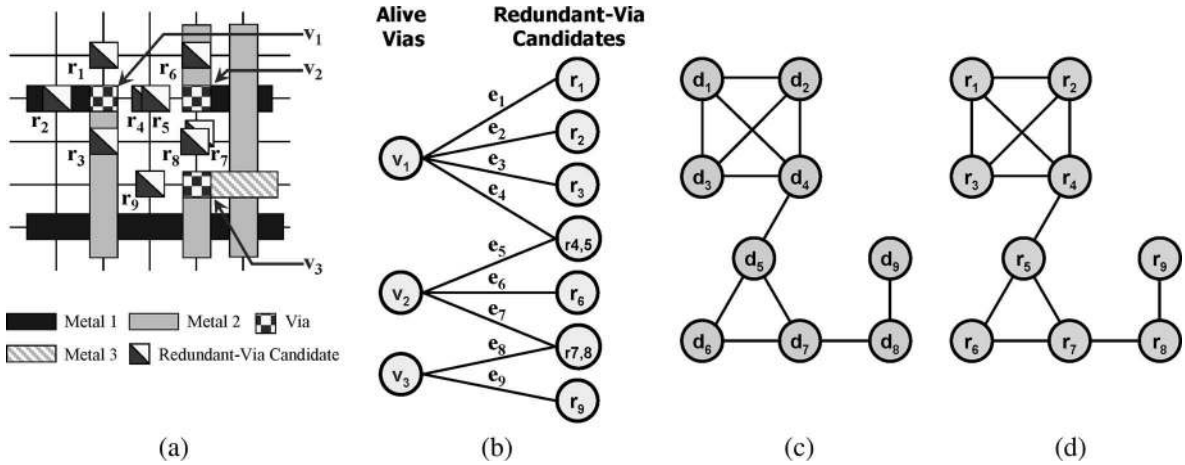

Fig. 12. Illustration of the proof for Theorem 1. (a) Postrouting layout. (b) Bipartite graph. (c) Line graph constructed from the bipartite graph. (d) Conflict graph constructed from the layout.

conflict exists between r_7 and r_8. Fig. 10(b) shows the cross-sectional view of the vertical design-rule conflict between r_7 and r_8. Fig. 10(c) gives the initial bipartite graph construction, and Fig. 10(d) presents the final bipartite graph after merging r_4 and r_5 into $r_{4,5}$ and merging r_7 and r_8 into $r_{7,8}$. For this modeling, a matching $(v_a, r_c) \in E$ in the bipartite graph will represent that an alive via $v_a \in V_A$ is paired with a redundant-via candidate $r_c \in V_C$ in the resulting routing layout. Notice that this approach readily extends to more general cases since the bipartite graph construction can be applied to both gridless and grid-based layouts.

The reason for the three-routing layers constraint is that, if we have a four-layer routing layout, the constructed bipartite graph may not lead to the optimal solution. For example, Fig. 11(a) shows a four-layer routing layout, and its bipartite graph is shown in Fig. 11(b). We can see that the optimal solution for double-via insertion is to insert a redundant via for each of vias v_1 and v_3, which cannot be obtained from the bipartite graph of Fig. 11(b).

We have the following lemmas and theorem showing the optimality of the bipartite graph formulation for grid-based routing with up to three routing layers and the stacked-via structure.

1) Definition 1: Conflict Graph: A conflict graph $C(V_G, E_G)$ is an undirected graph constructed from a detailed routing solution. A vertex $r \in V_G$ corresponds to a redundant-via candidate in the layout. An edge $(r_i, r_j) \in E_G$ exists iff both r_i and r_j are redundant-via candidates of the same via, or r_i and r_j will cause design-rule conflicts if they both exist.

2) Definition 2: Line Graph: The line graph of a graph G, which is denoted by $L(G)$, is a simple graph whose vertices are the edges of G, and (u, u') is an edge of $L(G)$ iff u and u' share a vertex of G.

Lemma 1: The PDVI problem can be reduced into a MIS problem.

Proof: In the conflict graph $C(V_G, E_G)$ of a routing layout, all vertices represent redundant-via candidates. The MIS of C, which is denoted by I, is a maximum vertex set such that, for any vertex pair $r_i, r_j \in I$, $(r_i, r_j) \notin E_C$. It means that all redundant-via candidates corresponding to vertices in I can be inserted by a redundant via at the same time without incurring any design-rule conflict, and for each via, only one of its redundant-via candidates is chosen for insertion. In addition, because the size of the independent set $|I|$ is the maximum, the number of inserted double vias is also the maximum. Therefore, the PDVI problem can be transformed to a MIS problem. ■

Lemma 2: Given a grid-based routing layout, the line graph of a bipartite graph formulating the PDVI problem, with up to three routing layers and the stacked-via structure, is isomorphic to the conflict graph of the routing layout. .

Proof: Given a bipartite graph $G_B(V_B, E_B)$ with the vertex set $V_B = V_A \cup V_C$, where V_A is the set of alive vias and V_C is the set of redundant-via candidates, we can construct its line graph $L(G_B) = (V_L, E_L)$. A vertex $d_i \in V_L$ corresponds to an edge $e_i \in E_B$. It is clear that each vertex $d_i \in V_L$ in $L(G_B)$ corresponds to a vertex r_i in the conflict graph $C(V_G, E_G)$. By Definition 2, an edge $(d_i, d_j) \in E_L$ exists iff, in G_B:

(1) e_i and e_j share the same vertex $v_k \in V_A$, or
(2) e_i and e_j share the same vertex $r_k \in V_C$.

It is clear that $(d_i, d_{i+1}, \ldots, d_p) \in E_L$ forms a clique if edges $(e_i, e_{i+1}, \ldots e_p)$ share the same vertex $v_k \in V_A$ or $r_k \in V_C$. For Case (1), it means that r_i and r_j are redundant-via candidates of the same single via, where r_i and r_j are redundant-via candidates incident by e_i and e_j, respectively. Thus, there also exists such an edge $(r_i, r_j) \in E_G$. Besides, redundant-via candidates of the same single via also form a clique in the conflict graph. For Case (2), it means that r_i and r_j have horizontal or vertical design-rule conflicts, and thus there also exists such an edge $(r_i, r_j) \in E_G$. For the redundant-via candidates in grid-based routing with up to three routing layers and the stacked-via structure, they also form a clique with conflict edges only. So the line graph induced from our bipartite graph is isomorphic to the conflict graph of the routing layout under the same assumption. See Fig. 12 for an example. ■

Lemma 3: A maximum matching in the formulated bipartite graph corresponds to a MIS in the conflict graph.

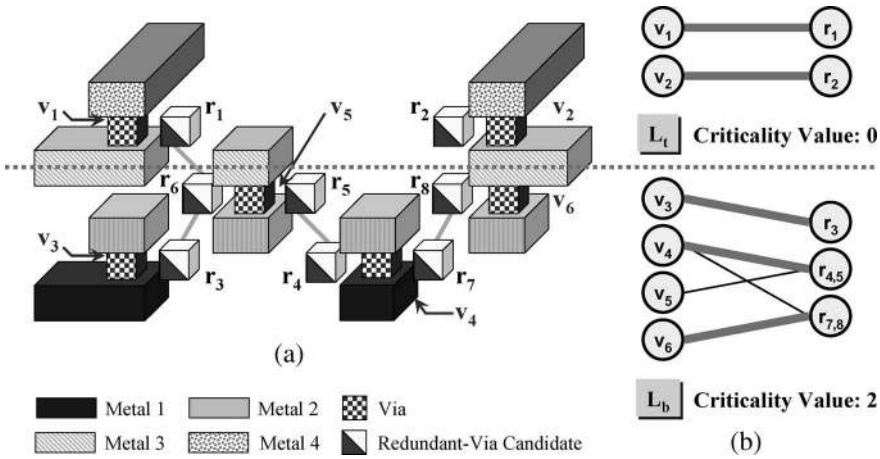

Fig. 13. Illustration of the TDVI algorithm. (a) Cross-sectional view of the whole layout. (b) Bipartite graphs of the subproblems.

Proof: Independent set in its line graph [23]: Thus, a maximum matching in G_B is an independent set in $L(G_B)$, in which the number of vertices is the maximum. Then, according to Lemma 2, $L(G_B)$ is isomorphic to conflict graph C. As a result, we can prove that a maximum matching solution in bipartite graph G_B is a MIS in conflict graph C. ■

Theorem 1: The maximum bipartite matching algorithm optimally solves the PDVI problem (i.e. the number of inserted double vias is maximum) for grid-based routing, with up to three routing layers (i.e., two via layers) and the stacked-via structure, in $O(\sqrt{|V|}|E|)$ time, where V is the set of vertices and E is the set of edges in the bipartite graph $G = (V, E)$.

Proof: By Lemmas 1–3 the PDVI problem with up to three routing layers can be exactly reduced into a maximum bipartite matching problem by constructing a bipartite graph from the given grid-based routing layout under the stacked-via structure. Let M denote a maximum matching in G_B. Then, a matching $(v_i, r_j) \in M$, where $v_i \in V_A$ and $r_j \in V_C$, represents that redundant-via candidate r_j is inserted for single via v_i. Through the bipartite maximum matching formulation, we can get a maximum matching solution, so that the number of double vias inserted is maximum, which is optimal. By applying Hopcroft and Karp's algorithm [23], a maximum matching could be found in $O(\sqrt{|V|}|E|)$ time. As a result, the theorem is established. ■

B. On-Track/Stacked Redundant-Via Enhancement

In the redundant-via insertion process, redundant vias can be placed *on-track* or *off-track*. A redundant via r of a via v is on-track if r is placed on the wire segment of v; it is off-track otherwise. In Fig. 10, redundant-via candidates r_2 and r_3 are on-track, whereas r_1 and r_4 are off-track. Since on-track redundant vias consume fewer routing resources than off-track ones, we prefer selecting on-track redundant-via candidates for insertion. Furthermore, we can also give higher priority to the redundant-via candidates of stacked vias to improve the reliability since stacked vias are more defect-prone and thus more desired for protection than single vias.

We formulate the PDVI problem considering on-/off-track redundant vias and stacked vias as a minimum weighted bipartite matching problem. The construction of vertices and edges is the same as Section IV-A, whereas the edge weight $w(v, r)$ of an edge $(v, r) \in E$ is given by

$$w(v,r) = \begin{cases} \frac{t_r}{n_s}, & \text{if } v \text{ is a stacked via} \\ & \quad \text{containing } n_s \text{ single vias} \\ t_r, & \text{otherwise} \end{cases}$$

$$t_r = \begin{cases} 1, & \text{if } r \text{ is on-track} \\ 2, & \text{if } r \text{ is off-track} \end{cases}.$$

With this weighting policy, the minimum weighted bipartite matching will give us a solution that prefers more on-track redundant vias and provides more protection for stacked vias.

C. TDVI Algorithm

For the general case with any number of routing layers, we propose a two-stage double-via insertion (TDVI) algorithm by extending the algorithm for up to three routing layers and the stacked-via structure.

First, we partition the original layout into sublayouts composed of up to three routing layers each, with the objective to minimize the number of vertical design-rule conflicts between sublayouts. Every redundant-via candidate r_c is associated with a *criticality* c_r. If r_c has a vertical design-rule conflict with some redundant-via candidates lying in the different sublayouts, then c_r is equal to the number of induced dead vias if r_c is inserted; c_r is equal to zero otherwise. The *criticality value* of a sublayout is the summation of the criticalities of the redundant-via candidates inside it. We then find solutions for the sublayouts one by one in the nondecreasing order of the criticality value. The sublayout with a lower criticality value has higher priority for processing since it contains more critical vias adjacent to the cut lines. During the subproblem solving stage, if we prefer selecting on-track/stacked redundant vias, the subproblem is formulated as a minimum weighted bipartite matching problem, as in Section IV-B; otherwise, it is solved by maximum bipartite matching, as in Section IV-A, to maximize the insertion rate. After solving one sublayout L_i, all sublayouts adjacent to L_i need to be updated by removing the infeasible redundant-via candidates. Continuing with the process, we can obtain the final solution for the original layout.

```
Algorithm: TDVI(L, V_C, on_track_stack_enh)
    Input : L - a resulting routing layout;
            V_C - a set of redundant-via candidates in L;
            on_track_stack_enh - a flag indexing if
                preferring on-track/stack redundant vias.
   Output : S - a solution for double-via insertion.
begin
1   S ← ∅;
2   if metal_layer(L) ≤ 3
3     if (on_track_stack_enh)
4       S ← MinimumWeightedBipartiteMatching(L);
5     else
6       S ← MaximumBipartiteMatching(L);
7   else
8     E_C is the set of vertical design-rule conflicts, E_C ← ∅;
9     for each pair r_i, r_j ∈ V_C with any vertical design-rule conflict
10      E_C ← E_C ∪ {(r_i, r_j)} ;
      /* Partitioning Stage */
11    Partition L into sublayouts {L_1, L_2, ..., L_n} by E_C such that
      metal_layer(L_i) ≤ 3 and # of conflicts between sublayouts
      is minimized;
      /* Calculating Criticality */
12    for each {(r_i, r_j) ∈ E_C | r_i and r_j lie in the different sublayouts}
13      Criticality c_{r_i} ← # of induced dead vias if r_i is inserted;
14      Criticality c_{r_j} ← # of induced dead vias if r_j is inserted;
15    for each sublayout L_i ∈ L
16      CV_{L_i} is the criticality value of L_i, CV_{L_i} ← Σ_{r∈L_i} c_r;
      /* Subproblem Solving Stage */
17    < L'_1, L'_2, ..., L'_n > ← sorted by non-decreasing CV_{L_i};
18    for L'_i = L'_1 to L'_n
19      if (on_track_stack_enh)
20        S_{L'_i} ← MinimumWeightedBipartiteMatching(L'_i);
21      else
22        S_{L'_i} ← MaximumBipartiteMatching(L'_i);
23      S ← S ∪ S_{L'_i};
24      Remove the infeasible redundant-via candidates;
25  return S;
end
```

Fig. 14. TDVI algorithm.

For the example shown in Fig. 13, the four-layer routing layout is partitioned into two sublayouts L_t and L_b with a vertical design-rule conflict between them. The redundant-via candidate r_1 in L_t has a vertical design-rule conflict, with r_6 lying in the different sublayout L_b, and there is no induced dead vias if we insert r_1. Therefore, c_1 is equal to 0. Similarly, r_6 in L_b has a vertical design-rule conflict, with r_1 lying in the different sublayout L_t, and two vias v_1 and v_3 will become dead vias if we insert r_6. Thus, c_6 is equal to 2. As a result, the criticality values of sublayouts L_t and L_b are equal to zero and two, respectively. Then, the two-layer sublayout L_t is processed before the three-layer sublayout L_b since L_t has lower criticality value than L_b. After getting the solution $\{(v_1, r_1), (v_2, r_2)\}$ for L_t, L_b is updated by removing the infeasible redundant-via candidate r_6. After getting the solution $\{(v_3, r_3), (v_4, r_4), (v_6, r_8)\}$ for L_b, the final solution for the whole layout is $\{(v_1, r_1), (v_2, r_2), (v_3, r_3), (v_4, r_4), (v_6, r_8)\}$. The TDVI algorithm is summarized in Fig. 14.

Note that, for the maximization of the redundant-via insertion rate, the TDVI algorithm is optimal for grid-based routing if no conflicts exist between the partitioned sublayouts, since each sublayout can be solved optimally with the maximum bipartite matching.

V. Experimental Result

The TBR algorithm has been implemented using the C++ programming language on a 1.2-GHz SUN Blade-2000 workstation with 8-GB memory.

We perform the routing algorithm on the MCNC benchmarks (provided by Cong *et al.* [10]) and the PDVI on both the MCNC and the industrial Faraday benchmarks (introduced in [1]). We consider the design rules of wire width, wire spacing, wire pitch, via width, and via spacing provided in these designs.

Tables I and II list the set of benchmark circuits. In the table, "Circuit" gives the names of the circuits, "Size" gives the layout dimensions in micrometer square, "#Layer" denotes the number of routing layers used, "#Net" gives the total number of nets, "#Connections" gives the number of two-pin connections after net decomposition, and "#Pin" gives the number of pins. The results can be downloaded from the web site http://cc.ee.ntu.edu.tw/~ywchang/research.html.

A. Routability-Driven Routing Framework

We compared TBR with three state-of-the-art gridless routers: 1) the Λ-shaped multilevel router MGR [8]; 2) the multilevel global routing + flat gridless detailed routing system (MARS) [10]; and 3) the V-shaped multilevel router (VMGR) [9]. MGR and VMGR were provided by Chen and Chang [8] and Chen *et al.* [9]. The experimental results on the single via count, routing completion rate, and runtime are listed in Table III. Here, the four routers were all performed in the routability-driven mode. MGR, VMGR, and TBR were run on the same machine, whereas the results of MARS were directly taken from [10]. (Note that [10] does not report the via count for MARS, and MARS was run on a 440-MHz SUN Ultra-10 workstation. We tried our best to make a fair comparison by normalizing its runtime by the factor 440/1200 based on the clock rates and reported the normalized results in Table III.) As shown in the table, all the four routers obtain 100% routing completion, whereas TBR achieves about 7.2×, 2.6×, and 1.4× runtime speedups compared with MGR, MARS, and VMGR, respectively. Furthermore, TBR reduces the single-via count by 20% and 23% over MGR and VMGR, respectively.

B. Double-Via Aware Detailed Routing

We also performed experiments on double-via aware gridless detailed routing. Table IV shows the routing results of TBR with and without double-via planning during detailed routing. In the table, "#Total Via" gives the total number of vias in the routing result, "#Dead Via" gives the number of dead vias, "#Critical Via" denotes the number of critical vias, and "#WL" reports the total wire length. The experimental results show that the redundant-via aware detailed routing results in fewer dead vias and critical vias by factors of 1.44 and 1.14, respectively, using similar running times. The slight increase in the via count (2%) is as expected, because the detailed router has to make detours not to incur more critical and dead vias. The small overhead on the via count also reflects the effectiveness of the via control by (3).

C. PDVI

We implemented the H2K and H3K algorithms proposed in [18] under the stacked-via structure, and named them H2K_S and H3K_S respectively. We compared our TDVI algorithm

TABLE I
MCNC BENCHMARK CIRCUITS

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

TABLE II
FARADAY BENCHMARK CIRCUITS

Circuit	Size (μm^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	36686	144703
RISC1	1003.6×1003.6	6	34034	95106	196677
RISC2	959.6×959.6	6	34034	95099	196670

TABLE III
COMPARISON FOR GRIDLESS ROUTERS

Circuit	MGR [8]			MARS [10]		VMGR [9]			TBR (Ours)		
	#Single Via	Cmp. Rate	Time (sec)	Cmp. Rate	Time (sec)	#Single Via	Cmp. Rate	Time (sec)	#Single Via	Cmp. Rate	Time (sec)
Mcc1	5791	100%	171.2	100%	38.8	5765	100%	51.5	5788	100%	25.7
Mcc2	33093	100%	3339.9	100%	702.9	34387	100%	1149.1	33153	100%	783.0
Struct	10055	100%	5.9	100%	11.6	9969	100%	3.5	7248	100%	2.8
Primary1	6374	100%	4.6	100%	12.3	6169	100%	4.2	5347	100%	2.7
Primary2	25955	100%	42.0	100%	59.7	24977	100%	23.7	22365	100%	17.7
S5378	8296	100%	41.0	100%	11.0	8520	100%	4.6	6784	100%	4.0
S9234	6833	100%	22.6	100%	8.4	6690	100%	3.4	5350	100%	2.9
S13207	17127	100%	122.6	100%	31.2	18263	100%	15.7	13767	100%	11.6
S15850	19875	100%	326.0	100%	39.3	23405	100%	20.0	16633	100%	16.4
S38417	50304	100%	362.8	100%	92.0	52798	100%	55.9	40655	100%	43.7
S38584	67026	100%	688.6	100%	170.9	74054	100%	191.6	54483	100%	148.5
Comp.	1.20	100%	7.15	100%	2.59	1.23	100%	1.37	1.00	100%	1.00

TABLE IV
REDUNDANT-VIA AWARE GRIDLESS ROUTING

Circuit	Without Redundant Via Consideration						With Redundant Via Consideration					
	Comp. Rate	#Total Via	#Dead Via	#Critical Via	WL (um)	Time (sec)	Comp. Rate	#Total Via	#Dead Via	#Critical Via	WL (um)	Time (sec)
Mcc1	100%	5371	616	1546	2.817E+07	25.7	100%	5501	388	1314	2.818E+07	29.8
Mcc2	100%	31229	4178	8739	4.055E+08	783.0	100%	32148	2617	7517	4.055E+08	997.3
Struct	100%	6746	58	664	8.726E+05	2.8	100%	7081	42	531	8.727E+05	3.1
Primary1	100%	5163	100	950	1.028E+06	2.7	100%	5312	63	800	1.028E+06	3.0
Primary2	100%	21754	586	4296	4.202E+06	17.7	100%	22306	435	3788	4.203E+06	20.0
S5378	100%	6416	619	1649	7.485E+04	4.0	100%	6387	445	1492	7.493E+04	5.0
S9234	100%	5121	436	1282	5.548E+04	2.9	100%	5129	340	1139	5.552E+04	3.4
S13207	100%	13223	1039	3246	1.786E+05	11.6	100%	13397	787	2913	1.789E+05	14.2
S15850	100%	15914	1400	3945	2.214E+05	16.4	100%	16120	1079	3682	2.217E+05	20.6
S38417	100%	39159	3294	9686	4.871E+05	43.7	100%	39345	2484	8795	4.879E+05	53.0
S38584	100%	52154	5605	13353	6.744E+05	148.5	100%	52894	3333	12185	6.756E+05	175.3
Comp.	1.00	0.98	1.44	1.14	1.00	0.84	1.00	1.00	1.00	1.00	1.00	1.00

with H2K_S, H3K_S, and a state-of-the-art commercial tool for post-layout double-via insertion.

Both H2K_S and H3K_S divide the original graph into subgraphs and apply an MIS solver to solve each subgraph. H2K_S can achieve a higher insertion rate than H3K_S ,while H3K_S is a modified heuristic of H2K_S and can increase the number of on-track redundant vias. Since H2K and H3K use *qualex-ms* [5] as their MIS solver, which is a Linux-based package, we performed the experiment on a Linux personal computer with an Intel Pentium 4 3.2-GHz central processing unit and 3-GB memory. The settings for the subgraph size (set to 1500) and the priority queue are the same as [18]. We ran TBR on the benchmarks to generate routing results, which were then fed into H2K_S, H3K_S, a commercial tool, and TDVI for PDVI. Our routing and double-via insertion results both passed the design rule check verification by the Cadence SOC Encounter.

Table V shows the double-via insertion comparison between TDVI and H2K_S. In the table, "Via Info." gives the total number of vias "#Total Via" and alive vias "#Alive Via"

TABLE V
COMPARISON FOR DOUBLE-VIA INSERTION WITH H2K

Circuit	Via Info.		H2K_S					TDVI (Ours)				
	#Total Via	#Alive Via	#Ins. Rvia	Ins. Rate	#On-Track Rvia	On-Track Rate	Time (s)	#Ins. Rvia	Ins. Rate	#On-Track Rvia	On-Track Rate	Time (s)
Mcc1	5501	5113	4932	96.5%	2318	47.0%	45.64	5011	98.0%	2100	41.9%	0.26
Mcc2	32148	29531	28670	97.1%	12982	45.3%	395.90	29099	98.5%	12360	42.5%	1.74
Struct	7081	7039	7019	99.7%	3001	42.8%	104.44	7037	99.9%	3444	48.9%	0.23
Primary1	5312	5249	5207	99.2%	2405	46.2%	73.08	5244	99.9%	2625	50.1%	0.16
Primary2	22306	21871	21713	99.3%	10242	47.2%	305.64	21842	99.9%	11294	51.7%	0.75
S5378	6387	5942	5611	94.4%	2724	48.5%	59.46	5806	97.7%	2658	45.8%	0.20
S9234	5129	4789	4543	94.9%	2271	50.0%	50.76	4710	98.4%	2113	44.9%	0.16
S13207	13397	12610	11983	95.0%	5970	49.8%	97.58	12356	98.0%	5543	44.9%	0.43
S15850	16120	15041	14272	94.9%	7033	49.3%	132.34	14718	97.9%	6726	45.7%	0.52
S38417	39345	36861	35093	95.2%	17469	49.8%	354.03	36170	98.1%	16188	44.8%	1.37
S38584	52894	49561	46823	94.5%	23044	49.2%	427.87	48478	97.8%	21954	45.3%	1.98
Comp.			0.98	96.4%	-	47.7%	299.26	1.00	98.6%	-	46.0%	1.00

TABLE VI
COMPARISON FOR DOUBLE-VIA INSERTION WITH H3K (TDVI RUNS WITH ON-TRACK/STACKED REDUNDANT-VIA ENHANCEMENT)

Circuit	Via Info.		H3K_S					TDVI (Ours)				
	#Total Via	#Alive Via	#Ins. Rvia	Ins. Rate	#On-Track Rvia	On-Track Rate	Time (s)	#Ins. Rvia	Ins. Rate	#On-Track Rvia	On-Track Rate	Time (s)
Mcc1	5501	5113	4908	96.0%	3494	71.2%	14.63	5012	98.0%	3759	75.0%	0.27
Mcc2	32148	29531	28650	97.0%	21698	75.7%	17.88	29101	98.5%	21955	75.4%	1.80
Struct	7081	7039	7029	99.9%	5552	79.0%	27.46	7037	99.9%	5912	84.0%	0.24
Primary1	5312	5249	5201	99.1%	3853	74.1%	37.84	5244	99.9%	4296	81.9%	0.16
Primary2	22306	21871	21677	99.1%	17370	80.1%	32.84	21842	99.9%	17803	81.5%	0.76
S5378	6387	5942	5571	93.8%	4181	75.0%	1.56	5806	97.7%	4468	77.0%	0.20
S9234	5129	4789	4559	95.2%	3228	70.8%	32.24	4710	98.4%	3724	79.1%	0.15
S13207	13397	12610	11944	94.7%	9258	77.5%	12.45	12356	98.0%	9843	79.7%	0.44
S15850	16120	15041	14214	94.5%	11072	77.9%	26.21	14718	97.9%	11559	78.5%	0.53
S38417	39345	36861	35091	95.2%	27621	78.7%	8.43	36170	98.1%	28741	79.5%	1.41
S38584	52894	49561	46817	94.5%	36314	77.6%	28.84	48478	97.8%	38208	78.8%	2.02
Comp.			0.98	96.3%	-	76.2%	70.82	1.00	98.6%	-	79.1%	1.00

TABLE VII
COMPARISON FOR DOUBLE-VIA INSERTION WITH A COMMERCIAL TOOL ON THE MCNC BENCHMARKS

Circuit	Via Info.		Commercial Tool			TDVI (Ours)		
	#Total Single Via	#Alive Single Via	#Ins. Rvia	Ins. Rate	Time (s)	#Ins. Rvia	Ins. Rate	Time (s)
Mcc1	5948	5471	4224	77.2%	10.0	5324	97.3%	0.2
Mcc2	34376	31212	26273	84.2%	146.0	30684	98.3%	1.0
Struct	7598	7543	5550	73.6%	19.0	7445	99.9%	0.2
Primary1	5536	5468	4214	77.1%	18.0	5417	99.1%	0.2
Primary2	23154	22663	17649	77.9%	76.0	22486	99.2%	0.8
S5378	6739	6235	5697	91.4%	12.0	6048	97.0%	0.2
S9234	5365	4991	4528	90.7%	10.0	4854	97.3%	0.2
S13207	13972	13096	11930	91.1%	48.0	12752	97.4%	0.4
S15850	16922	15693	14280	91.0%	64.0	15256	97.2%	0.5
S38417	40942	38235	34625	90.6%	318.0	37223	97.4%	1.4
S38584	55381	51602	46846	90.8%	540.0	50133	97.2%	2.0
Comp.			0.87	85.0%	125.41	1.00	97.9%	1.00

for the routing results, "#Ins. Rvia" shows the number of inserted double vias after the insertion process, "Ins. Rate" is equal to "#Ins. Rvia" divided by "#Alive Via," "#On-Track Rvia" represents the number of on-track double vias, and "On-Track Rate" is equal to "#On-Track Rvia" divided by "#Ins. Rvia." Compared with H2K_S, on average, TDVI obtains 299.3× runtime speedup and achieves a higher insertion rate at 98.6%.

With the on-track/stacked redundant-via enhancement, we compared TDVI with H3K_S. Table VI shows the results. Compared with H3K_S, on average, TDVI obtains 70.8× runtime speedup and achieves a higher insertion rate at 98.6% with the on-track insertion rate at 79.1%. The runtime improvement matches our expectation because the bipartite matching enjoys the polynomial-time complexity, whereas the MIS formulation is NP-complete. Thus, H2K_S and H3K_S need more running times for achieving high-quality solutions.

Tables VII and VIII show the double-via insertion results compared with the commercial tool for the MCNC and the Faraday benchmarks, respectively. For the MCNC benchmark, on average, TDVI can obtain 125.4× runtime speedup and achieve a higher insertion rate at 98.6%; for the Faraday benchmark, TDVI can also obtain 6.1× runtime speedup and achieve a higher insertion rate at 99.80%.

TABLE VIII
COMPARISON FOR DOUBLE-VIA INSERTION WITH A COMMERCIAL TOOL ON THE FARADAY BENCHMARKS

Circuit	Via Info.		Commercial Tool			TDVI (Ours)		
	#Total Single Via	#Alive Single Via	#Ins. Rvia	Ins. Rate	Time (sec)	#Ins. Rvia	Ins. Rate	Time (sec)
DMA	111047	71331	71192	99.81%	28.00	71331	100.00%	4.74
DSP1	225911	133449	132128	99.01%	71.00	133198	99.81%	11.34
DSP2	223684	131820	130719	99.16%	66.00	131546	99.79%	10.76
RISC1	342259	203055	199947	98.47%	117.00	202487	99.72%	19.51
RISC2	348005	207590	204262	98.97%	119.00	206919	99.68%	19.94
Comp.			0.99	98.97%	6.05	1.00	99.80%	1.00

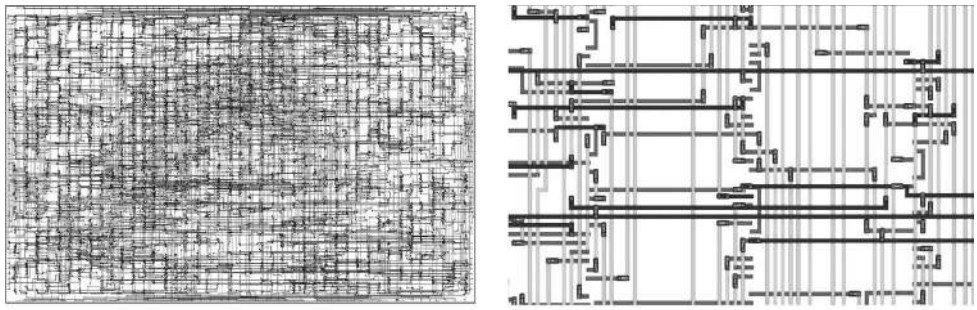

Fig. 15. Double-via insertion result for "S13207" obtained from TDVI. (a) Full-chip view (double vias are highlighted as black circles). (b) Local view.

Notice that, in Table V, the insertion rates obtained by TDVI for all circuits, except "Mcc1" and "Mcc2," are optimal, because these designs contain only three routing layers. For "Mcc1," it is also reasonable that the number "#Ins. Rvia" in Table VI is more than that in Table V, because the insertion process is not optimal for designs with more than three routing layers. Fig. 15(a) and (b) shows the PDVI result for "S13207" with full-chip and local views, respectively.

VI. CONCLUSION

We have presented a new two-pass bottom-up full-chip gridless router, named TBR, considering double-via insertion for yield enhancement. We have also proposed an optimal polynomial-time PDVI algorithm for grid-based routing with up to three routing layers and the stacked-via structure and have extended the algorithm to handle the general problem. Experimental results have shown the effectiveness and efficiency of the proposed methods.

Future work lies in performing wire perturbation or spreading and/or extending wires to allocate additional space for each dead via to further improve the insertion rate. To add more redundant vias for each alive via, furthermore, we may iteratively perform our TDVI algorithm, i.e., we first insert the first redundant via for each alive via, then find an additional redundant-via candidate in the updated layout and insert the second redundant via, and so on.

ACKNOWLEDGMENT

The authors would like to thank the Editor, the Associate Editor, and the reviewers for the very prompt handling and the very constructive comments.

REFERENCES

[1] S. N. Adya, S. Chaturvedi, J. A. Roy, D. Papa, and I. L. Markov, "Unification of partitioning, floorplanning and placement," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 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] G. A. Allan, "Targeted layout modifications for semiconductor yield/reliability enhancement," *IEEE Trans. Semicond. Manuf.*, vol. 17, no. 4, pp. 573–581, Nov. 2004.

[4] U. Brenner and A. Rohe, "An effective congestion-driven placement framework," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 4, pp. 387–394, Apr. 2003.

[5] *QUALEX package*. [Online]. Available: http://www.busygin.dp.uanpc.html

[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] H.-Y. Chen, M.-F. Chiang, Y.-W. Chang, L. Chen, and B. Han, "Novel full-chip gridless routing considering double-via insertion," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2006, pp. 755–760.

[8] T.-C. Chen and Y.-W. Chang, "Multilevel gridless routing considering optical proximity correction," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Jan. 2005, pp. 1160–1163.

[9] T.-C. Chen, Y.-W. Chang, and S.-C. Lin, "A novel framework for multilevel full-chip gridless routing," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Jan. 2006, pp. 636–641.

[10] J. Cong, J. Fang, M. Xie, and Y. Zhang, "MARS–A multilevel full-chip gridless routing system," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 3, pp. 382–394, Mar. 2005.

[11] J. Cong, J. Fang, and Y. Zhang, "Multilevel approach to full-chip gridless routing," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2001, pp. 396–403.

[12] 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.

[13] 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.

[14] Y.-L. Hsieh and T.-M. Hsieh, "A new effective congestion model in floorplan design," in *Proc. IEEE/ACM Des., Autom. Test Eur. Conf.*, Mar. 2004, pp. 1204–1209.

[15] R. M. Karp, "Reducibility among combinatorial problems," in *Proc. Symp. Complexity Computer Comput.*, 1972, pp. 85–103.

[16] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern routing: Use and theory for increasing predictability and avoiding coupling," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 7, pp. 777–790, Nov. 2002.
[17] C. Y. Lee, "An algorithm for path connection and its application," *IRE Trans. Electron. Comput.*, vol. 10, pp. 346–365, 1961.
[18] K.-Y. Lee and T.-C. Wang, "Post-routing redundant via insertion for yield/reliability improvement," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Jan. 2006, pp. 303–308.
[19] 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.
[20] J. Lou, S. Thakur, S. Krishnamoorthy, and H. S. Sheng, "Estimating routing congestion using probabilistic analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 1, pp. 32–41, Jan. 2002.
[21] M. Marek-Sadowska, "Global router for gate array," in *Proc. IEEE Int. Conf. Comput. Des.*, Oct. 1984, pp. 332–337.
[22] *Taiwan Semiconductor Manufacturing Company (TSMC)*. Reference Flow 5.0 and Reference Flow 6.0.
[23] D. B. West, *Introduction to Graph Theory*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2001.
[24] J. G. Xi, *Improving Yield in RTL-to-GDSII Flows*. Manhasset, NY: EE Times, Jul. 11, 2005.
[25] J. Xiong and L. He, "Probabilistic congestion model considering shielding for crosstalk reduction," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Jan. 2005, pp. 739–742.
[26] G. Xu, L.-D. Huang, D. Z. Pan, and M. D. Wong, "Redundant-via enhanced maze routing for yield improvement," in *Proc. IEEE/ACM Asia South Pacific Des. Automa. Conf.*, Jan. 2005, pp. 1148–1151.
[27] H. Yao, Y. Cai, X. Hong, and Q. Zhou, "Improved multilevel routing with redundant via placement for yield and reliability," in *Proc. ACM Great Lakes Symp. VLSI*, 2005, pp. 143–146.

Huang-Yu Chen (S'05) received the B.S. degree in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, R.O.C., in 2004. He is currently working toward the Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan. His research interests include physical design for VLSI circuits and manufacturability-driven large-scale routing.

Mr. Chen was the recipient of the 2006 Outstanding Research Award from the National Taiwan University and the Best Paper Nomination from ICCAD 2007.

Mei-Fang Chiang (S'05) received the B.S. degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, R.O.C., in 2004 and the M.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2006. She is currently working toward the Ph.D. degree in the Graduate School of Information, Production and Systems, Waseda University, Kitakyushu, Japan.

Her current research interests include the design for manufacturability.

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

He is currently a Professor with the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University. His current research interests include physical design for VLSI circuits, design for manufacturability, and design automation for biochips.

Lumdo Chen received the B.S. and M.S. degrees in electrical engineering from National Chiao Tung University, Hsinchu, Taiwan R.O.C., in 1982 and 1986, respectively.

He has been with United Microelectronic Corporation (UMC), Hsinchu, for 21 years. He was a Microcontroller Design Manager in the late 1980s and a Corporate CAD Manager in the 1990s. He is currently the Chairman of the EDA committee with UMC.

Brian Han received the B.S. degrees in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, R.O.C., in 1985, and the M.S. degree in electrical engineering from Polytechnic University, Brooklyn, NY, in 1989.

He has been with several companies both in the USA and Taiwan for 17 years. The major fields he works on are CAD tool development and ASIC design. He is currently the Department Manager of the CAD Group, United Microelectronic Corporation, Hsinchu, and mainly focuses on design methodology and design automation.