Novel Full-Chip Gridless Routing Considering Double-Via Insertion [*]

Huang-Yu Chen[1], Mei-Fang Chiang[2], Yao-Wen Chang[1,2],
Lumdo Chen[3], and Brian Han[3]
[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan
[2]Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan
[3]United Microelectronics Corporation, Hsinchu 300, Taiwan

ABSTRACT

As the technology node advances into the nanometer era, via-open defects are one of the dominant failures. 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 post-layout stage. The increasing design complexity, however, leaves very limited space for post-layout optimization. It is thus desirable to consider the double-via insertion at both routing and post-routing 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. We also propose a new post-layout double-via insertion algorithm to achieve a higher insertion rate. Based on a bipartite graph matching formulation, we develop an optimal double-via insertion algorithm for the cases with up to three routing layers and the stack-via structure, and then extend the algorithm to handle the general cases. Experiments show that our methods significantly improve the via count, the number of dead vias, double-via insertion rates, and running times.

Categories and Subject Descriptors: B.7.2 [Integrated Circuits]: Design Aids - Layout, Placement and Routing

General Terms: Algorithms, Designs, Reliability

Keywords: Manufacturability, redundant via insertion, routing

1. INTRODUCTION

As IC process geometries shrink to 90nm 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. The failure significantly reduces the manufacturing yield and chip performance.

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 [15], double vias lead to 10X–100X smaller failure rates than single vias. Existing approaches are often for post-layout optimization by replacing a single via with a double-via structure as long as it does not create any design-rule violations [1, 11]. The increasing design complexity, however, leaves very limited space for post-layout optimization. It has been reported that inserting redundant vias during routing can improve the double-via insertion rates by 15–25% than the post-layout optimization, at the cost of routability degradation. Therefore, it is desired to consider the double-via insertion at both routing and post-routing stages for better trade-off in routability and double-via insertion rates.

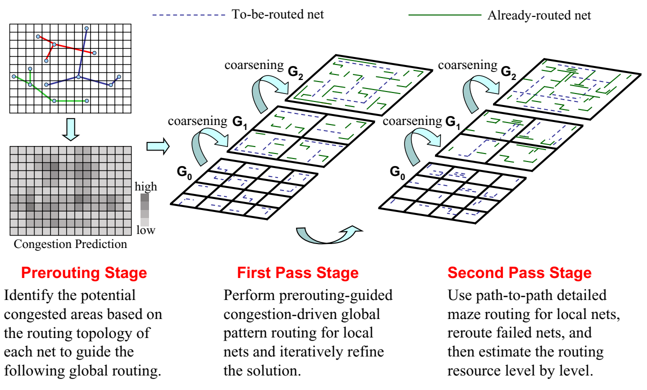
Xu *et al.* [17] proposed a pioneering work to consider double-via insertion during maze routing. By assigning double-via costs to the routing graph, they formulated the problem as a multi-objective maze routing problem and applied Lagrangian relaxation to solve it. However, they only consider the redundant via at the detailed routing stage, their cost modeling for the number of vias is not accurate enough, and the high time complexity of Lagrangian relaxation limits the feasible problem size to be within hundreds of nets. Yao *et al.* [18] developed a grid-based router which features via-minimization global routing followed by double-via aware detailed routing; the work claimed that the post-layout double-via insertion problem can be solved by a maximum bipartite matching formulation, which was recently shown to be incorrect for some cases in [11]. Lee and Wang [11] instead formulated the double-via insertion problem as a maximum independent set (MIS) problem. Since the MIS problem for a general graph is NP-complete, they resorted to heuristics to handle the problem.

The continuously increasing design complexity imposes severe challenges for modern routers. 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 [6, 7], MR [3], CMR [8], and MGR [4]) while the V-shaped one consists of top-down *uncoarsening* followed by bottom-up *coarsening* (*e.g.*, VMGR [5]). 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 [8] and optical proximity correction (OPC) [4]. 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) [3, 10]. 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 [5].

In this paper, we present a new full-chip gridless routing system, named TBR (Two-pass Bottom-up gridless Router), 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* rout-

[*]This work was supported in part by NSC of Taiwan under Grant No's. NSC 94-2215-E-002-005 and NSC 94-2752-E-002-008-PAE, TSMC, and UMC.

ing stage, and then followed by a bottom-up *detailed* routing stage. See Figure 1 for an illustration of 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 [3, 10]. 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.



Prerouting Stage
Identify the potential congested areas based on the routing topology of each net to guide the following global routing.

First Pass Stage
Perform prerouting-guided congestion-driven global pattern routing for local nets and iteratively refine the solution.

Second Pass Stage
Use path-to-path detailed maze routing for local nets, reroute failed nets, and then estimate the routing resource level by level.

Figure 1: The new two-pass, bottom-up routing framework.

We also propose a new post-layout double-via insertion algorithm to achieve a higher insertion rate. Based on a bipartite graph matching formulation, we develop an optimal double-via insertion algorithm for the cases with up to three routing layers and the stack-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 1.20X compared with the state-of-the-art gridless router [4, 5, 6], and our redundant-via aware detailed router can effectively obtain fewer dead vias by 1.41X. Additionally, compared with the state-of-the-art double-via insertion algorithm [11], our post-layout double-via insertion algorithm can achieve 98.6% double-via insertion rate with at least 70.8X runtime speedup.

The rest of this paper is organized as follows. Section 2 describes the routing model and the post-layout redundant-via insertion problem. Section 3 presents our novel two-pass, bottom-up routing framework considering redundant-via insertion. Section 4 presents our post-layout double-via insertion algorithm. Experimental results are reported in Section 5, and conclusions are given in Section 6.

2. PRELIMINARIES

2.1 Global Routing Model

We model the routing resource as a *routing graph*. 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.

2.2 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 [6], with a modification to guarantee the correctness of the searched path. Figure 2 (a) gives a detailed routing instance with the source s and target t, and Figure 2 (b) is 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 Figure 2 (b) represent the routable and unroutable nodes, respectively.

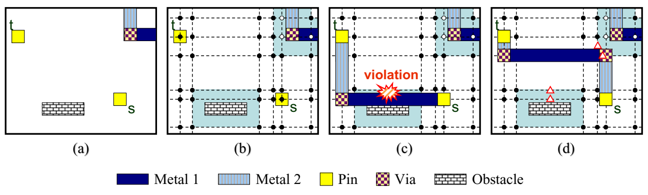


Figure 2: The gridless detailed routing model. (a) A detailed routing instance. (b) The implicit connection graph. (c) An infeasible detailed routing path. (d) A design-rule-correct detailed routing path.

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 [6]. As shown in Figure 2 (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.* [4, 5] added a horizontal and a 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. Figure 2 (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.

2.3 Two-Pass Routing Framework Model

As illustrated in Figure 1, G_k corresponds to the routing graph of the 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 the level k). After global routing of level k is performed, we merge four GC_k's into a larger GC_{k+1} and at the same time perform resource estimation for use at 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 path-to-path detailed maze routing is performed and rip-up/re-route procedures are applied for failed nets. The process continues until we reach the coarsest level when the final routing solution is obtained.

2.4 Post-Layout Double-Via Insertion

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. 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 *stack via*. A stack via is the via consisting of at least two vertically stacked single vias.

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

- Problem *PDVI* (*Post-layout Double-Via Insertion*): Given a post-routing 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.

3. 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 post-layout double-via insertion process.

To deal with the simultaneous optimization, we propose a novel two-pass, bottom-up routing framework which 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 Figure 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, TBR consists of (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 subsections.

3.1 Congestion-Driven Prerouting Stage

To improve routability, the works [3, 4, 5] 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, TBR features a prerouting stage that identifies the potentially congested areas based on the routing topology of each net. With the prerouting, TBR can perform global routing and detailed routing separately 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 2-pin connections, with each connection corresponding to an edge of the MST. TBR then pre-estimates the congestion in the routing graph for all 2-pin connections using the *probabilistic congestion model* which has recently been successfully applied to placement [2], floorplanning [9], and routing [12, 16] and is generally believed to have the ability to alleviate the net-ordering problem in sequential routing. TBR pre-evaluates the congestion as the average number of global 1-bend and 2-bend routes that might pass through the boundary of adjacent GCs. For a 2-pin connection c, we first explore all possible 1- and 2-bend global routes from its source s to its target t, denoted by the 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 $N_c(i) = \{r \in P_c \mid r \text{ is the route passing through } b_i\}$, then the estimated congestion of b_i with respect to c equals $|N_c(i)|/|P_c|$. For example, as shown in Figure 3 (a), the connection c has five possible 1- and 2-bend routes from the source s to target t. Figure 3 (b) gives the number of routes passing through each global cell boundary b_i, $|N_c(i)|$, and Figure 3 (c) shows the congestion estimation of c in the routing graph.

3.2 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 [10]. 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 $R_c = \{e \in E_0 \mid e \text{ is the}$

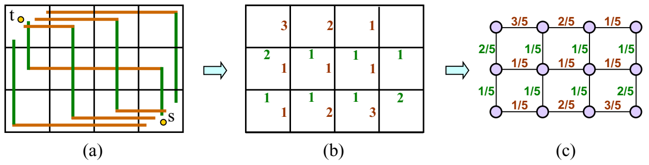


Figure 3: Probabilistic congestion estimation. (a) Two 1-bend and three 2-bend routes from s to t. (b) The number of routes through each boundary. (c) The pre-estimation congestion in the routing graph.

edge chosen for routing$\}$. For the congestion control, we define the cost function of the global routing result R_c as follows:

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

where α, 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 (1-bend) or Z-shaped (2-bend) route to make the connection, which gives the shortest path length between two points. Therefore, the wire length is minimum, and 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 the density d_e in Equation (2) comes from both the predicted congestion obtained at the prerouting stage and real routing. Its value is dynamically updated as the routing proceeds.

3.3 Redundant-Via Aware Detailed Routing Stage

Same as 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 post-layout redundant-via insertion in this stage. To do so, Xu *et al.* [17] considered via minimization and redundant-via planning during detailed routing. They assigned redundant-via costs to 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 consider redundant-via planning and via minimization simultaneously, we define the following cost function for a net n to guide the maze routing:

$$cost(n) = \gamma V_n + \delta P_n, \tag{3}$$

where γ and δ are user-defined parameters, V_n is the number of vias in n, and P_n is a redundant-via related penalty function for n.

We explain the determination of P_n. We call a redundant-via candidate as 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 Figure 4 (a)). Since a safe via can always be protected by its safe neighbor, we set the cost 0 to all its redundant-via candidates. The *degree of freedom* of a via v, denoted by DoF_v, is the total number of redundant-via candidates of v, as defined in [17]. If via v is not a safe via, we assign the cost $1/DoF_v$ to its redundant-via candidates. The rational is that if via v becomes a dead via due to the route n (*i.e.*, n passes all the redundant-via candidates of v), the cost for routing n would be increased by $DoF_v \times \frac{1}{DoF_v} = 1$, which exactly equals the number of the induced dead via. Note that a redundant-via candidate r_v may be shared by more than one via. In this case, we set the cost of r_v as $\{\max\{1/DoF_{v_i}\} \mid v_i$ is the via that shares $r_v\}$. Figure 4 (b) shows the redundant-via candidates and their cost assignments. Finally, the penalty function P_n of net n is the summation of the costs of the redundant-via candidates that are passed through by n. The P_n of the route n in Figure 4 (c) is calculated as $1/3 + 1/2 + 1/3 = 7/6$.

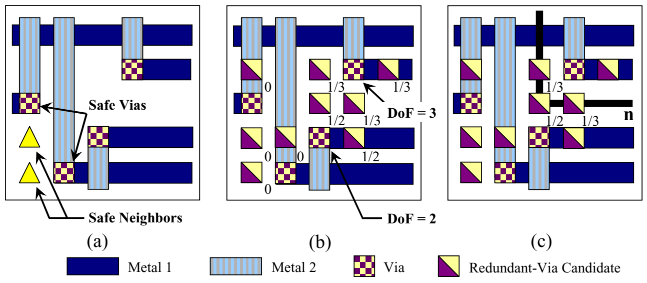


Figure 4: Redundant-via related penalty assignment. (a) Safe neighbors and safe vias. (b) Cost assignments. (c) The penalty function of the route n equals 7/6.

Let the number of dead vias induced by a net n be $\Phi(n)$. We have the following theorem for the redundant-via related penalty function.

THEOREM 1. $\Phi(n) \leq P_n$.

4. POST-LAYOUT DOUBLE-VIA INSERTION

For the PDVI (Post-layout Double-Via Insertion) problem, we develop a polynomial-time optimal double-via insertion algorithm for the case with up to three routing layers (*i.e.*, two via layers) and the stack-via structure (*i.e.*, two or more vias vertically stacked are treated as one stack via; see Figure 5 (a)), based on a bipartite graph matching formulation. With the optimal 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 [11] that cannot be solved by the method presented in [18] can be formulated exactly as shown in Figure 5 (b).

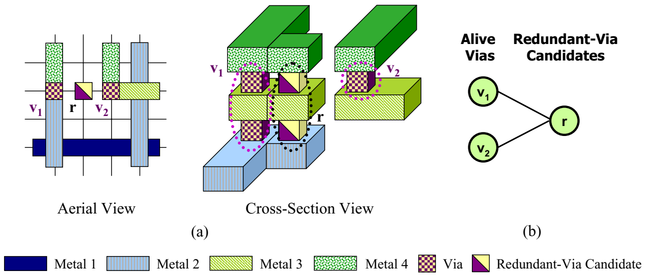


Figure 5: (a) Stack-via structure. (b) Bipartite graph of (a).

4.1 Optimal Algorithm for up to Three Routing Layers

Our bipartite graph construction is as follows. Given a resulting routing layout, 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 $v \in V_A$ and $r \in V_C$, an edge $(v, r) \in E$ exists if r is a redundant-via candidate of v. 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, Figure 6 (a) is a resulting gridless routing layout, where a horizontal conflict exists between r_4 and r_5 and a vertical conflict exists between r_7 and r_8. See Figure 6 (b) for the cross-section view of the vertical design-rule conflict between r_7 and r_8. Figure 6 (c) gives the initial bipartite graph construction, and Figure 6 (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 $(v, r) \in E$ in the bipartite graph will represent that an alive via $v \in V_A$ is paired with a redundant via $r \in V_C$ 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.

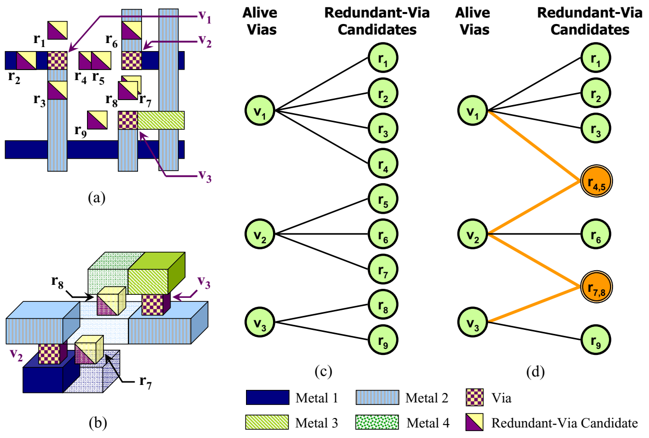


Figure 6: Bipartite graph construction. (a) A gridless routing layout. (b) A cross-section view for the vertical design-rule conflict between r_7 and r_8. (c) The initial bipartite graph without considering conflicts. (d) The final bipartite graph after merging r_4, r_5 and r_7, r_8.

We have the following lemmas and theorem for the optimality of our bipartite graph formulation.

LEMMA 1. *The line graph [14] of a bipartite graph formulating the PDVI problem is isomorphic to the conflict graph [14] under the stack-via structure constraint.*

LEMMA 2. *A maximum matching in the formulated bipartite graph corresponds to a maximum independent set in the conflict graph.*

THEOREM 2. *The maximum bipartite matching algorithm optimally solves the PDVI problem (i.e. the number of double via inserted is maximum) with up to three routing layers (i.e., two via layers) and the stack-via structure in $O(\sqrt{V}E)$ time, where V is the number of vertices and E is the number of edges in the bipartite graph.*

4.2 On-Track/Stack 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 Figure 6, 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. Further, we can also give higher priority to the redundant-via candidates of stack vias to improve the reliability since stack 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 stack vias as a minimum weighted bipartite matching problem. The construction of vertices and edges are the same as Section 4.1, while the edge weight $w(v, r)$ of an edge $(v, r) \in E$ is given by

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

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

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

4.3 Two-Stage Double-Via Insertion 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 stack-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 is associated with a *criticality* c_r. If r has a vertical design-rule conflict with some redundant-via candidates lying in the different sublayouts, then c_r equals the number of induced dead vias if r is inserted; c_r equals 0, otherwise. The *criticality value* of a sublayout is the summation of the criticalities of redundant-via candidates inside it. We then find solutions for the sublayouts one by one in the non-decreasing 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/stack redundant vias, the subproblem is formulated as a minimum weighted bipartite matching problem as in Section 4.2; otherwise, it is solved by maximum bipartite matching as in Section 4.1 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.

For the example shown in Figure 7, the 4-layer layout is partitioned into 2 sublayouts L_t and L_b with a vertical conflict between them. The criticalities of r_1 and r_6 are equal to 1 and 2, respectively. Then the 2-layer sublayout L_t is processed before the 3-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 Figure 8.

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

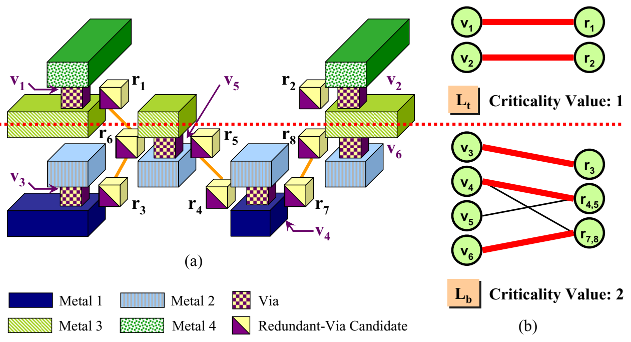


Figure 7: An illustration of *TDVI* algorithm. (a) The cross-section view of the whole layout. (b) Bipartite graphs of the subproblems.

```
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
  $S \leftarrow \emptyset$;
  if $metal\_layer(L) \leq 3$
    if ($on\_track\_stack\_enh$)
      $S \leftarrow MinimumWeightedBipartiteMatching(L)$;
    else
      $S \leftarrow MaximumBipartiteMatching(L)$;
  else
    $E_C$ is the set of vertical design-rule conflicts, $E_C \leftarrow \emptyset$;
    for each pair $r_i, r_j \in V_C$ with any vertical design-rule conflict
      $E_C \leftarrow E_C \cup \{(r_i, r_j)\}$ ;
    /* Partitioning Stage */
    Partition $L$ into sublayouts $\{L_1, L_2, ..., L_n\}$ by $E_C$ such that
    $metal\_layer(L_i) \leq 3$ and # of conflicts between sublayouts
    is minimized;
    /* Calculating Criticality */
    for each $\{(r_i, r_j) \in E_C \mid r_i$ and $r_j$ lie in the different sublayouts$\}$
      Criticality $c_{r_i} \leftarrow$ # of induced dead vias if $r_i$ is inserted;
      Criticality $c_{r_j} \leftarrow$ # of induced dead vias if $r_j$ is inserted;
    for each sublayout $L_i \in L$
      $CV_{L_i}$ is the criticality value of $L_i$, $CV_{L_i} \leftarrow \sum_{r \in L_i} c_r$;
    /* Subproblem Solving Stage */
    $< L'_1, L'_2, ..., L'_n > \leftarrow$ sorted by non-decreasing $CV_{L_i}$;
    for $L'_i = L'_1$ to $L'_n$
      if ($on\_track\_stack\_enh$)
        $S_{L'_i} \leftarrow MinimumWeightedBipartiteMatching(L'_i)$;
      else
        $S_{L'_i} \leftarrow MaximumBipartiteMatching(L'_i)$;
      $S \leftarrow S \cup S_{L'_i}$;
      Remove the infeasible redundant-via candidates;
  return $S$;
end
```

Figure 8: TDVI Algorithm.

5. EXPERIMENTAL RESULTS

The TBR algorithm has been implemented using the C++ programming language on a 1.2 GHz SUN Blade-2000 workstation with 8 GB memory. The 11 experimental routing benchmarks are provided by the authors of [6] and are commonly used in this community.

We compared TBR with three state-of-the-art gridless routers: the Λ-shaped multilevel router MGR [4], the multilevel global routing+flat gridless detailed routing system (MARS) [6], and the V-shaped multilevel router (VMGR) [5]. MGR and VMGR were provided by the authors of [4, 5]. The experimental results on the single via count, routing completion rate, and runtime are listed in Table 1. Here the four routers were all performed in the routability-driven mode. MGR, VMGR, and TBR were run on the same machine while the results of MARS were directly taken from [6]. (Note that [6] 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 1.) As shown in the table, all the four routers obtain 100% routing completion, while TBR achieves about 7.2X, 2.6X, and 1.4X runtime speedups compared with MGR, MARS, and VMGR, respectively. Further, TBR reduces the single-via count by respective 20% and 24% over MGR and VMGR.

We also performed experiments on double-via aware gridless detailed routing. Table 2 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 and "#Critical Via" denotes the number of critical vias. The experimental results show that the redundant-via aware detailed routing results in fewer dead vias and critical vias by the respective factors 1.41X and 1.14X 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 Equation (3).

We compared our TDVI algorithm with H2K and H3K algorithms proposed in [11] for post-layout double-via insertion. Both H2K and H3K divide the original graph into subgraphs and use an MIS solver to solve each subgraph. H2K can achieve higher insertion rate than H3K while H3K is a modified heuristic of H2K and can increase the number of on-track redundant vias. Since H2K and H3K use *qualex-ms* [13] as their MIS solver, which is a Linux-based package, we performed the experiment on a Linux PC with an Intel Pentium 4 3.2 GHz CPU and 3 GB memory. The settings for the subgraph size (set to 1500) and the priority queue are the same as [11]. We ran TBR on the benchmarks to generate routing results, which were then fed into H2K, H3K, and TDVI for post-layout double-via insertion. Our routing and double-via insertion results are both passed the Design Rule Check (DRC) verification by the Cadence SOC Encounter.

Table 3 shows the double-via insertion comparison between TDVI and H2K. In the table, "Via Info." gives the total number of vias "#Total Via" and alive vias "#Alive Via" 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" equals "#On-Track Rvia" divided by "#Ins. Rvia". Compared with H2K, on average TDVI obtains 299.3X runtime speedup and achieves a higher insertion rate at 98.6%.

With the on-track/stack redundant via enhancement, we compared TDVI with H3K. Table 4 shows the results. Compared with H3K, on average TDVI obtains 70.8X runtime speedup and achieves a higher insertion rate at 98.6% with the on-track insertion rate at 79.2%. The runtime improvement is as we expected because the bipartite matching enjoys polynomial-time complexity, whereas the MIS problem is NP-complete. Thus H2K and H3K need more running times for achieving high solution quality.

Notice that in Table 3, 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 4 is more than that in Table 3 because the insertion process is not optimal for designs with more than three routing layers.

Table 1: Comparison for gridless routers.

Circuit	MGR [4] #Single Via	MGR [4] Cmp. Rate	MGR [4] Time (*s*)	MARS [6] Cmp. Rate	MARS [6] Time (*s*)	VMGR [5] #Single Via	VMGR [5] Cmp. Rate	VMGR [5] Time (*s*)	TBR (Ours) #Single Via	TBR (Ours) Cmp. Rate	TBR (Ours) Time (*s*)
Mcc1	5791	100%	171.2	100%	38.8	5765	100%	51.5	5687	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%	7.2	100%	2.6	1.24	100%	1.4	1	100%	1

6. 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 post-layout double-via insertion algorithm for the cases with up to three routing layers and the stack-via structure, and have extended the algorithm to handle the general problem. Experimental results have shown the effectiveness and efficiency of the proposed methods.

Table 2: Redundant-via aware gridless detailed routing.

Circuit	Without Redundant Via Consideration Comp. Rate	#Total Via	#Dead Via	#Critical Via	Time (s)	With Redundant Via Consideration Comp. Rate	#Total Via	#Dead Via	#Critical Via	Time (s)
Mcc1	100%	5385	616	1546	25.7	100%	5508	388	1313	29.8
Mcc2	100%	31237	4179	8741	783.0	100%	32153	2617	7518	997.3
Struct	100%	6763	58	664	2.8	100%	7097	42	531	3.1
Primary1	100%	5176	100	950	2.7	100%	5325	63	800	3.0
Primary2	100%	21788	586	4298	17.7	100%	22338	435	3789	20.0
S5378	100%	6448	619	1656	4.0	100%	6411	445	1498	5.0
S9234	100%	5144	434	1284	2.9	100%	5158	338	1140	3.4
S13207	100%	13263	1037	3248	11.6	100%	13443	786	2913	14.2
S15850	100%	15969	1396	3950	16.4	100%	16179	1077	3695	20.6
S38417	100%	39289	3287	9702	43.7	100%	39489	2477	8804	53.0
S38584	100%	52295	4509	13358	148.5	100%	53057	3330	12185	175.3
Comp.	1.00	0.98	1.41	1.14	0.84	1	1	1	1	1

Table 3: Comparison for double-via insertion with H2K.

Circuit	Via Info. #Total Via	Via Info. #Alive Via	H2K [11] #Ins. Rvia	H2K [11] Ins. Rate	H2K [11] #On-Track Rvia	H2K [11] On-Track Rate	H2K [11] Time (s)	TDVI (Ours) #Ins. Rvia	TDVI (Ours) Ins. Rate	TDVI (Ours) #On-Track Rvia	TDVI (Ours) On-Track Rate	TDVI (Ours) Time (s)
Mcc1	5508	5120	4932	96.3%	2318	47.0%	45.6	5027	98.2%	2172	43.2%	0.26
Mcc2	32153	29536	28670	97.1%	12982	45.3%	395.9	29124	98.6%	12524	43.0%	1.74
Struct	7097	7055	7019	99.5%	3001	42.8%	104.4	7053	99.9%	3444	48.8%	0.23
Primary1	5325	5262	5207	99.0%	2405	46.2%	73.1	5258	99.9%	2630	50.0%	0.16
Primary2	22338	21903	21713	99.1%	10242	47.2%	305.6	21872	99.9%	11275	51.5%	0.75
S5378	6411	5966	5611	94.0%	2724	48.5%	59.5	5830	97.7%	2664	45.7%	0.20
S9234	5158	4820	4543	94.3%	2271	50.0%	50.8	4741	98.4%	2119	44.7%	0.16
S13207	13443	12657	11983	94.7%	5970	49.8%	97.6	12403	98.0%	5541	44.7%	0.43
S15850	16179	15102	14272	94.5%	7033	49.3%	132.3	14773	97.8%	6734	45.6%	0.52
S38417	39489	37012	35093	94.8%	17469	49.8%	354.0	36319	98.1%	16198	44.6%	1.37
S38584	53057	49727	46823	94.2%	23044	49.2%	427.9	48632	97.8%	21967	45.2%	1.98
Comp.			0.98	96.1%	-	47.7%	299.3	1	98.6%	-	46.1%	1

Table 4: Comparison for double-via insertion with H3K (TDVI runs with on-track/stack redundant via enhancement).

Circuit	Via Info. #Total Via	Via Info. #Alive Via	H3K [11] #Ins. Rvia	H3K [11] Ins. Rate	H3K [11] #On-Track Rvia	H3K [11] On-Track Rate	H3K [11] Time (s)	TDVI (Ours) #Ins. Rvia	TDVI (Ours) Ins. Rate	TDVI (Ours) #On-Track Rvia	TDVI (Ours) On-Track Rate	TDVI (Ours) Time (s)
Mcc1	5508	5120	4908	95.9%	3494	71.2%	14.6	5028	98.2%	3802	75.6%	0.27
Mcc2	32153	29536	28650	97.0%	21698	75.7%	17.9	29118	98.6%	22056	75.7%	1.80
Struct	7097	7055	7029	99.6%	5552	79.0%	27.5	7053	99.9%	5929	84.1%	0.24
Primary1	5325	5262	5201	98.8%	3853	74.1%	37.8	5258	99.9%	4309	82.0%	0.16
Primary2	22338	21903	21677	99.0%	17370	80.1%	32.8	21872	99.9%	17831	81.5%	0.76
S5378	6411	5966	5571	93.4%	4181	75.0%	1.6	5830	97.7%	4485	76.9%	0.20
S9234	5158	4820	4559	94.6%	3228	70.8%	32.2	4741	98.4%	3749	79.1%	0.15
S13207	13443	12657	11944	94.4%	9258	77.5%	12.5	12403	98.0%	9878	79.6%	0.44
S15850	16179	15102	14214	94.1%	11072	77.9%	26.2	14773	97.8%	11594	78.5%	0.53
S38417	39489	37012	35091	94.8%	27621	78.7%	8.4	36319	98.1%	28858	79.5%	1.41
S38584	53057	49727	46817	94.1%	36314	77.6%	28.8	48632	97.8%	38322	78.8%	2.02
Comp.			0.97	96.0%	-	76.2%	70.8	1	98.6%	-	79.2%	1

7. REFERENCES

[1] G. A. Allan, "Targeted Layout Modifications for Semiconductor Yield/Reliability Enhancement," *IEEE TSM*, vol. 17, Nov. 2004.

[2] U. Brenner and A. Rohe, "An Effective Congestion-Driven Placement Framework," *IEEE TCAD*, vol. 22, No. 4, pp. 387–394, Apr. 2003.

[3] Y.-W. Chang and S.-P. Lin, "MR: A New Framework for Multilevel Full-Chip Routing," *IEEE TCAD*, vol. 23, no. 5, pp. 793–800, May 2004.

[4] T.-C. Chen and Y.-W. Chang, "Multilevel Gridless Routing Considering Optical Proximity Correction," *Proc. ASP-DAC*, pp. 1160–1163, Jan. 2005.

[5] T.-C. Chen, Y.-W. Chang, and S.-C. Lin, "A Novel Framework for Multilevel Full-Chip Gridless Routing," *Proc. ASP-DAC*, pp. 636–641, Jan. 2006.

[6] J. Cong, J. Fang, M. Xie, and Y. Zhang, "MARS–A Multilevel Full-Chip Gridless Routing System," *IEEE TCAD*, vol. 24, no. 3, pp. 382–394, Mar. 2005.

[7] J. Cong, J. Fang, and Y. Zhang, "Multilevel Approach to Full-Chip Gridless Routing," *Proc. ICCAD*, pp. 396–403, Nov. 2001.

[8] T.-Y. Ho, Y.-W. Chang, S.-J. Chen, and D.-T. Lee, "Crosstalk- and Performance-Driven Multilevel Full-Chip Routing," *IEEE TCAD*, vol. 24, no. 6, pp. 869–878, Jun. 2005.

[9] Y.-L. Hsieh and T.-M. Hsieh, "A New Effective Congestion Model in Floorplan Design," *Proc. DATE*, vol. 2, pp. 1204–1209, Feb. 2004.

[10] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern Routing: Use and Theory for Increasing Predictability and Avoiding Coupling," *IEEE TCAD*, pp. 777–790, Nov. 2002.

[11] K.-Y. Lee and T.-C. Wang, "Post-Routing Redundant Via Insertion for Yield/Reliability Improvement," *Proc. ASP-DAC*, pp. 303–308, Jan. 2006.

[12] J. Lou, S. Thakur, S. Krishnamoorthy, and H. S. Sheng, "Estimating Routing Congestion Using Probabilistic Analysis," *IEEE TCAD*, vol. 21, no. 1, pp. 32–41, Jan. 2002.

[13] QUALEX package, http://www.busygin.dp.ua/npc.html

[14] D. B. West, *Introduction to Graph Theory*, Prentice Hall, 2nd Ed., 2001.

[15] J. G. Xi, "Improving Yield in RTL-to-GDSII Flows," *EE Times*, Jul. 11, 2005.

[16] J. Xiong and L. He, "Probabilistic Congestion Model Considering Shielding for Crosstalk Reduction," *Proc. ASP-DAC*, pp. 739–742, Jan. 2005.

[17] G. Xu, L.-D. Huang, D. Z. Pan, and M. D. Wong, "Redundant-Via Enhanced Maze Routing for Yield Improvement," *Proc. ASP-DAC*, pp. 1148–1151, Jan. 2005.

[18] H. Yao, Y. Cai, X. Hong, and Q. Zhou, "Improved Multilevel Routing with Redundant Via Placement for Yield and Reliability," *Proc. GLSVLSI*, pp. 143–146, 2005.