Optimizing the Antenna Area and Separators in Layer Assignment of Multilayer Global Routing

Wen-Hao Liu and Yih-Lang Li, *Member, IEEE*

***Abstract*—Traditional solutions to antenna effect, such as jumper insertion and diode insertion performed at post-route stage may produce extra vias and degrade circuit performance. Previous work suggests combining layer assignment, jumper insertion, and diode insertion together to achieve a better design quality with less additional cost. Based on our observations on global and local antenna violations, this paper proposes an antenna-safe single-net layer assignment (AS-SLA), which first enumerates all antenna-safe layer assignment solutions of a net, and then extracts the minimum-cost one for the net. AS-SLA can minimize via count and separators as well. In addition, an antenna avoidance layer assignment flow (AALA) adopting AS-SLA as its kernel not only avoids global antenna violations, but also eliminates local antenna violations. Experimental results reveal that, in 16 benchmarks, AALA can yield ten antenna-violation-free assignments, while the algorithms of other works yield no antenna-violation-free assignment. However, AALA performs about seven times slower than other antenna-aware layer assignment algorithm. Accordingly, two acceleration techniques are proposed to reduce the runtime of AALA by 57.6%.**

***Index Terms*—Antenna effect, design for manufacturability, global routing, layer assignment, separator.**

I. Introduction

THE ANTENNA effect is a relevant topic that must be considered in the routing stage to improve circuit reliability. The antenna effect is charge collection by plasma during lithography, possibly resulting in gate oxide damages and ultimately the circuit reliability problem. When exposed to plasma, a wire segment functioning as an antenna may gather charging current. If the wire segment connects only to the gate oxides (but not diffusions) and collects significantly more charges than a threshold value does, Fowler-Nordheim tunneling current discharges through the thin oxide to the substrate, incurring the gate oxide damage [3], [4]. For instance, Fig. 1(a) shows the side view of a routing result connecting a driver (diffusion) to a gate, wire segment e_4 located on metal 1, e_1, e_3, and e_5 located on metal 2, and e_2 located on metal 3. Fig. 1(b) shows the circumstance, while metals 1 and 2 are etched. Since metal 3 has not been built yet, the collected charges of e_3, e_4, and e_5 discharge through the thin gate oxides. In Fig. 1(b), wire segments e_3, e_4, and e_5 function as an antenna. Also, the gate oxide damage may occur if the collected charges of the antenna exceed a threshold, which is referred to as antenna violation. Fig. 1(c) shows the circumstance after manufacturing metal 3, in which the collected charges can be released through the diffusion.

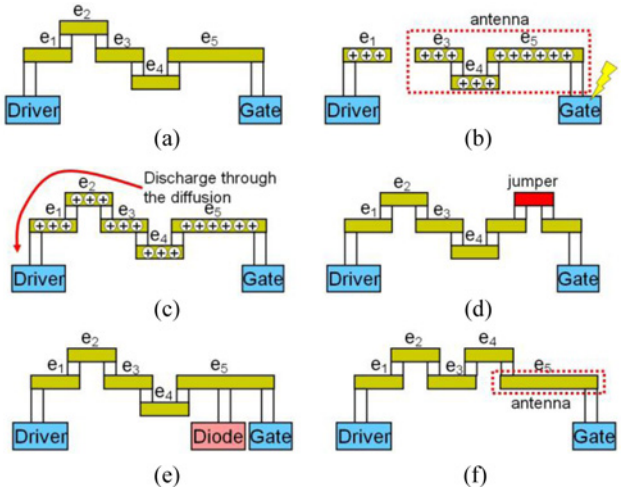


Fig. 1. (a) Side view of a routing result connecting a driver to a gate. (b) Circumstance in which metal 1 and metal 2 are etched. (c) Circumstance after manufacturing metal 3. (d) Solving antenna violation by jumper insertion. (e) Solving antenna violation by diode insertion. (f) Routing result as layer assignment obeys the antenna rule.

Jumper insertion and diode insertion are generally adopted to solve the antenna violation problem during detailed routing or post optimization stages. Fig. 1(d) illustrates an example of solving the antenna violation problem by using jumper insertion [5]. The wire segments with antenna violations are split and then routed to the top-metal layer; the wire on the top-metal layer is called a jumper. The jumper cuts a long antenna into a shorter one, yet consumes additional vias. On the other hand, by placing diodes near the gates with antenna violations [Fig. 1(e)], the inserted diodes can protect the gates from current discharge through the thin oxide to the substrate by restraining the charge voltage level. However, jumper insertion and diode insertion consume additional vias and routing resources to fix antenna problem, which may degrade the circuit performance and manufacturing yield.

In contrast to repairing antenna violations during the post-route stage, considering the antenna effect during the global

Manuscript received March 23, 2013; revised September 22, 2013; accepted November 2, 2013. Date of current version March 17, 2014. This work was supported in by the National Science Council of Taiwan under Grant NSC 102-2220-E-009–027. This paper was recommended by Associate Editor E. Young.

The authors are with the Department of Computer Science, National Chiao-Tung University, Hsinchu, Taiwan (e-mail: dnoldnol@gmail.com; ylli@cs.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2013.2293053

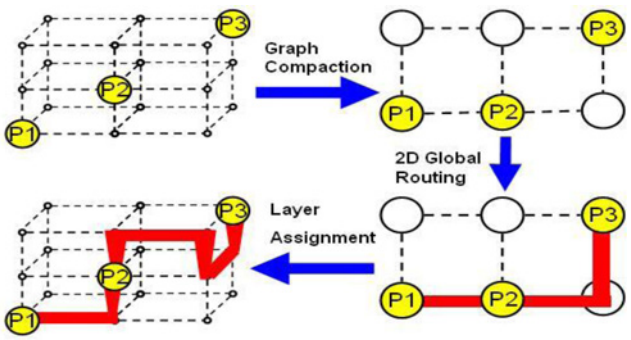


Fig. 2. Modern 3-D global routing flow.

routing stage can prevent antenna violations at a lower cost and may reduce the effort of the detailed routing and post optimization stages for fixing antenna violations. Fig. 2 displays the common global routing flow adopted by most state-of-the-art global routers [6]–[15]. This flow first condenses the 3-D grid graph into a 2-D grid graph, and then obtains the 2-D routing result via 2-D global routing algorithms. In the final stage, layer assignment assigns each net edge to its corresponding metal layer to obtain the final 3-D global routing results. In [1], considering the antenna effect in the layer assignment stage is regarded as capable of effectively reducing antenna violations.

The layer assignment problem in the global routing stage typically focuses on minimizing the via count without altering the routing topology or increasing congestion, which is called constrained via count minimization problem and has been proven to be NP-complete [16]. The constrained via count minimization problem has been studied in the works of [13], [17]–[19], and some investigations [20], [21] have further considered the timing issue in this problem. To reduce antenna violations, the works of [1] and [2] extend layer assignment algorithms to take antenna rules into account. Fig. 1(f) shows a routing result as layer assignment complies with the antenna rule. Although Fig. 1(a) and (f) have the same via count, the antenna area in Fig. 1(f) is smaller than that in Fig. 1(a). Using antenna-aware layer assignment in the global routing stage can reduce the antenna violations happened in the detailed routing stage. As a result, the cost and effort of jumper insertion and diode insertion can ease off.

The works of [1] and [2] applied a tree-partitioning algorithm to facilitate antenna avoidance layer assignment. Those algorithms treat each net as a tree, and break each into several sub-trees by tree-partitioning algorithms. The wire segments of each sub-tree are then assigned to their corresponding layers conforming to the antenna rule. Although efficient, these tree-partitioning-based algorithms may fall into the local optimal. This work presents an antenna avoidance layer assignment flow called antenna avoidance layer assignment flow (AALA), capable of avoiding antenna violations and minimizing the via count at the same time. Experimental results reveal that the proposed algorithm provides a more global view and achieve a higher quality than previous tree-partitioning-based algorithms.

The rest of paper is organized as follows. Section II describes the problem formulation. Section III introduces previous works. Next, Section IV then presents an antenna-safe single-net layer assignment algorithm (AS-SLA) and two acceleration techniques for AS-SLA. Section V presents AALA that adopts AS-SLA to assign each single net. Additionally, Section VI summarizes the experimental results. Conclusions are finally drawn in Section VII.

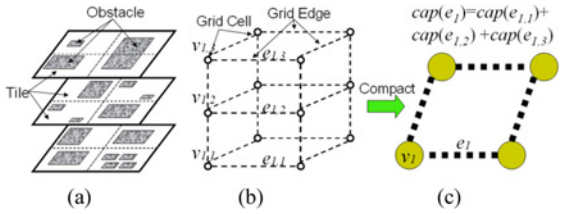


Fig. 3. Mapping a 3-D routing region to the grid graph model. (a) Three-layer routing region. (b) 3-D grid graph. (c) Compacted 2-D grid graph from (b).

II. Problem Formulation

A. Preliminaries

The grid graph model is generally applied in the global routing and layer assignment problems. According to Fig. 3(a), a k-layer routing region can be partitioned into an array of tiles and modeled by a k-layer grid graph $G^k(V^k, E^k)$ such as that shown in Fig. 3(b), where V^k denotes the set of 3-D grid cells, each grid cell represents a tile, and E^k refers to the set of 3-D grid edges, in which each grid edge is termed by the adjacency of the related tile of its two end nodes. The capacity of a grid edge e [$cap(e)$] indicates the number of wire segments that can pass through e. The overflow of a grid edge e is defined as the amount of demand in excess of capacity.

The layer assignment problem is formulated as follows. Given a 3-tuple(G^k, G, S), $G^k(V^k, E^k)$ denotes a k-layer 3-D grid graph; $G(V, E)$ denotes a 2-D grid graph that is compressed from G^k; and S refers to a 2-D global routing result on G. The 3-D grid edge $e_{i,z}$, $1 \le z \le k$, is called the corresponding edge of 2-D grid edge e_i, and the 3-D grid node $v_{i,z}$, $1 \le z \le k$, is called the corresponding node of 2-D grid node v_i. Layer assignment assigns the wire segments of S to the corresponding edges in order to obtain a 3-D global routing result S^k. For instance, in Figs. 3(b) and (c), $e_{1,1}$, $e_{1,2}$ and $e_{1,3}$ are the corresponding edges of e_1; by assuming that a 2-D global router identifies a wire segment passing through e_1 in Fig. 3(c), layer assignment assigns the wire segment to one of $e_{1,1}$, $e_{1,2}$ and $e_{1,3}$ in Fig. 3(b). With the imposed wire congestion constraints and antenna rule, layer assignment can ensure the feasibility of assignment results.

B. Wire Congestion Constraints

Given a 2-D global routing result S, layer assignment identifies the k-layer assignment result S^k. To ensure its legality and routability, S^k must satisfy the following wire congestion constraints:

$$TWO(S^k) \le TWO(S) \tag{1}$$

$$MWO(S^k) \le \lceil MWO(S) \times (2/k) \rceil \tag{2}$$

where TWO and MWO denote the total wire overflow and maximum wire overflow, respectively. The first constraint

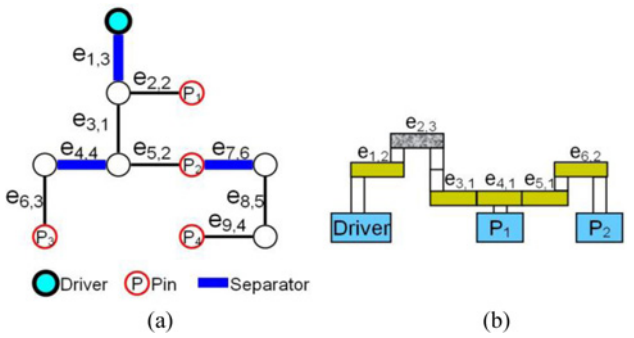


Fig. 4. (a) 3-D global routing solution of a net. (b) Example with local-antenna-violation.

ensures that the wire overflow in 3-D graph does not exceed that in 2-D graph. Each routing layer has its preferable routing direction in the benchmarks used herein, and the numbers of horizontal and vertical layers in these benchmarks are equal. Thus, the second constraint ensures that the peak congestion of an edge in 2-D graph can be uniformly distributed to its corresponding edges in 3-D graph on the layers of the same preferable routing direction as that of the edge in 2-D graph.

C. Antenna Rule

Given a 3-D global routing result of a net with a driver and a set of output pins (gates). For each output pin p, let $L_{top}(p)$ denote the top layer containing at least one wire segment in the routing path between the driver and p. The antenna of p is defined as the maximum sub-tree that starts at p and terminates each tree branch growth whenever meeting a leaf of the entire tree or a wire segment whose layer is larger than or equal to $L_{top}(p)$. The wire segments surrounding antennas are called separators [1]. For instance, Fig. 4(a) shows a 3-D routing result of a net, the notation next to each wire segment denotes the 3-D grid edge that the wire segment is assigned. The path from the driver to pin p_1 contains wire segments $e_{1,3}$ and $e_{2,2}$, so $L_{top}(p_1)$ is 3; the antenna of p_1 is thus a sub-tree consisting of wire segments $\{e_{2,2}, e_{3,1}, e_{5,2}\}$, and the layer of each wire segment is lower than 3. Additionally, the wire segments $e_{1,3}$, $e_{4,4}$, and $e_{7,6}$ are separators because they terminate the tree growth of the antenna of p_1. For other pins in Fig. 4(a), $L_{top}(p_2)$, $L_{top}(p_3)$, and $L_{top}(p_4)$ are 3, 4, and 6, respectively. The antennas of p_2, p_3, and p_4 include the wire segment sets, $\{e_{2,2}, e_{3,1}, e_{5,2}\}$, $\{e_{6,3}\}$, and $\{e_{8,5}, e_{9,4}\}$, respectively. Notably, p_1 and p_2 share the same antenna. The blue wire segments in Fig. 4(a) are separators. A separator is formally defined as follows.

Definition 1: Separator: a wire segment $e_{i,j}$ is regarded as a separator if $e_{i,j}$ is the nearest wire segment to pin p in a path from p to the root such that the layer of $e_{i,j}$ is $L_{top}(p)$.

To avoid gate oxide damage, the work in [2] targets to make the antenna ratio of each antenna less than a given threshold A_{max}. The work in [2] adopts the cumulative antenna model to define the antenna ratio of an antenna as follows:

$$\text{antennaratio} = \frac{\text{exposedareaoftheantenna}}{\text{totalgateoxidearea}} \tag{3}$$

where the total gate oxide area is the sum of the gate oxide area of the pins connecting to the antenna. To simplify the antenna ratio calculation, [2] assumed a uniform wire width and gate oxide area. Thus, the antenna ratio calculation can be simplified as L/n, where L denotes the antenna length and n denotes the number of output pins connected by this antenna. Moreover, in [2], an antenna violation is said to occur as L/n of an antenna is larger than A_{max}, so each antenna is purposefully designed to conform to the constraint, $L/n \leq A_{max}$, during layer assignment. In Fig. 4(a), the antenna ratio of the associated antennas of p_1, p_2, p_3, and p_4 are 1.5, 1.5, 1, and 2, respectively.

Complying with the antenna rule of [2] can reduce the antenna violations. However, in some circumstances, layer assignment results that obey the abovementioned antenna rule still damage gate oxides. For instance, $e_{2,3}$ is the separator of p_1 and p_2 [Fig. 4(b)] and assume that A_{max} is 2. The p_1 and p_2 share an antenna $\{e_{3,1}, e_{4,1}, e_{5,1}, e_{6,2}\}$. The antenna ratio of $\{e_{3,1}, e_{4,1}, e_{5,1}, e_{6,2}\}$ is 2, which does not exceed A_{max}, explaining why Fig. 4(b) conforms to the above antenna rule. However, during manufacturing, while metal 1 is etched and metals 2 and 3 have not yet been built, the collected charges of e_3, e_4 and e_5 may discharge through p_1 and then damage the gate oxide of p_1. This circumstance is referred to herein as local-antenna-violation. Conversely, the circumstance in which the antenna ratio of an entire sub-tree exceeds A_{max} is defined to global-antenna-violation. The work in [2] addressed the feasibility of eliminating global-antenna-violation, yet was unaware of local-antenna-violation. Thus, the experiment in Section VI reveals that the layer assignment results of [2] contain a significant amount of antenna-violations. In this paper, local-antenna-violation is avoided using a strict antenna rule, i.e., $L \leq A_{max}$, where L denotes the antenna length. If an assignment result of a net conforms to this strict antenna rule, the assignment result is regarded as antenna-safe. This finding implies that the net never incurs global-antenna-violation and local-antenna-violation.

D. Objectives

In this paper, minimizing the number of nets with antenna violations is of priority concern, while minimizing the via count is the secondary objective. Moreover, the fact that the separators must be fixed at a specified layer explains why too many separators may degrade the flexibility of the subsequent detailed routing. Thus, minimizing separators is the third objective. Additionally, the final assignment result has to obey wire congestion constraints. In this paper, minimizing antenna violations is regarded as an objective rather than a constraint, since the remaining antenna violations in the assignment result can be resolved by jumper or diode insertion at the detailed routing stage.

III. Previous Works

Section III-A briefly describes the layer assignment works in [13] and [17]–[19] that focuses on minimizing the via count under wire congestion constraints, yet fails to consider the antenna effect. Section III-B then introduces current antenna

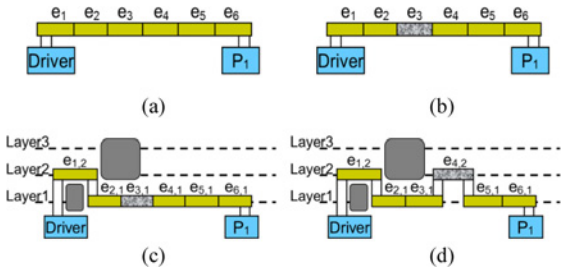


Fig. 5. Comparison between the work of [2] and this paper. (a) 2-D routing result of a net. (b) Separator location determined by a tree-partitioning algorithm. (c) Assignment result with antenna violations obtained by [2]. (d) Antenna-safe assignment result obtained by this paper.

avoidance layer assignment works in [1] and [2], and discusses the potential limitations of [1] and [2].

A. Layer Assignment for Via Count Minimization

The works in [13], [17], and [18] determined the assignment order of nets first, and then assigned each single net sequentially under wire congestion constraints. Although their methods are efficient, the layer assignment quality is limited by the net order. To obey wire congestion constraints, the nets in the later assigning order have less available layer resources than those assigned in the early assigning order. As a result, the layer assignment solutions for the later assigning nets are bad. Accordingly, how to determinate a good assignment order is a critical issue addressed in [13], [17], and [18].

The author of [19] proposes a negotiation-based via count minimization (NVM) framework to overcome the available resource problem for the nets in the later assigning order. NVM first identifies a minimal via count solution without considering wire congestion constraints for each net, and then iteratively rips-up and reassigns the nets with overflows to gradually meet wire congestion constraints.

In [19], a single-net layer assignment algorithm called NANA was developed, which can always identify the minimum-cost assignment result for a net. Reference [19] formulates vias and congestion into the objective cost function of NANA to optimize these two issues both. NANA is a two-phase dynamic-programming algorithm. During the first phase, NANA enumerates all possible 3-D trees starting at leaf nodes in a bottom-up manner until reaching the root. During the second phase, the minimum-cost assignment solution for the entire tree is extracted from a set of possible 3-D trees. Each net edge is then assigned to the corresponding layer based on the minimum-cost assignment solution in a top-down manner.

B. Antenna Avoidance Layer Assignment

The work in [1] presents a two-step algorithm to solve the single-net layer assignment problem with the antenna rule but does not consider the congestion issue, while the work in [2] extends the two-step algorithm to assign each single net one by one for solving the antenna avoidance layer assignment problem under wire congestion constraints.

Given a net, the first step of the two-step algorithm used in [1] and [2] determines the location of separators by a tree-partitioning algorithm. The separators decompose a net into several sub-nets, subsequently causing the antenna ratio of each sub-net below or equal to A_{max}. In the second step, each separator is assigned to a layer, and then the wire segments of each sub-net are assigned to the layers lower than the surrounding separators. Note that, the second step of the single-net layer assignment in [2] avoids assigning separators and wire segments to the layers that may violate wire congestion constraints. As a result, the second step may identify no solution or an inferior solution with many vias since the wire congestion and via count are not considered in the first step. For example, assuming A_{max} is 3 and given a 2-D routing result of a net as shown in Fig. 5(a), the two-step algorithm used in [2] first adopts the tree-partitioning algorithm to determine e_3 to be a separator and decompose the net into two sub-nets; the length of each sub-net is less than or equal to 3 [Fig. 5(b)]. After that, the separator and other wire segments are assigned to the corresponding layers [Fig. 5(c)]. The gray rectangles in Fig. 5(c) denote congested regions. As the regions at layers 2 and 3 are congested, and [2] regards wire congestion constraints as hard constraints, separator e_3 cannot be assigned to higher layer. As a result, the separator cannot cut the associated antenna of p_1 into shorter one, the antenna violation occur.

In contrast to the two-step algorithm used in [2], the proposed AS-SLA can determine the separator locations and assigning wire segments to the corresponding layers in a single step. The via count, wire congestion and antenna rule can be optimized simultaneously as well. For the case of Fig. 5(a), AS-SLA can identify the antenna-safe result as shown in Fig. 5(d). Additionally, previous works [2] lack the ability to control the number of separators, while this paper formulates the separator cost into the objective function of AS-SLA in order to control the number of separators.

IV. Antenna-Safe Single-Net Layer Assignment

If a net has at least an antenna-safe assignment solution, the proposed AS-SLA can identify the minimum-cost antenna-safe solution for this net. AS-SLA is similar to NANA [19], they are both based on a dynamic-programming method. However, NANA does not consider the antenna effect. Section IV-A details the algorithm of AS-SLA. Section IV-B presents two acceleration techniques for AS-SLA.

A. Algorithm Flow of AS-SLA

The single-net layer assignment problem is formulated as follows. Given a 2-D routing result of a net $N(V_N, E_N)$ where V_N and E_N denote the sets of 2-D grid nodes and 2-D grid edges passed by N, respectively; and given a set of pins of N in G^k, which is denoted by P_N^k. The net N can be regarded as a 2-D tree, and the root and the leaves of the tree must contain a pin. The single-net layer assignment problem identifies a 3-D tree in G^k to connect all pins of P_N^k. As the layer assignment result of N, this 3-D tree consists of a set of 3-D grid edges, denoted as E_N^k, the edges of which E_N^k are the corresponding edges of E_N. For instance, a single net layer assignment reads a 2-D routing result $N(V_N, E_N)$ in Fig. 6(a) and a set of pins P_N^k in Fig. 6(b), $V_N = \{v_0, v_1, v_2, v_3, v_4, v_5\}$, $E_N = \{e_1, e_2, e_3,$

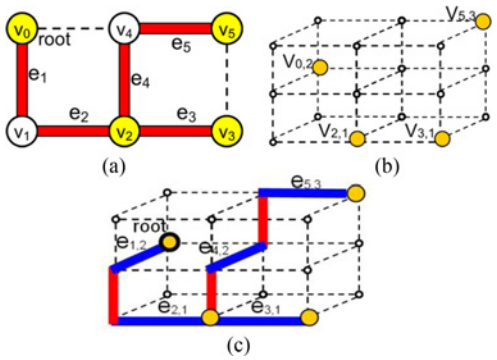


Fig. 6. Example of a single net layer assignment. (a) 2-D routing result of a net $N(V_N, E_N)$. (b) Set of N's pin locations in G^k. (c) Layer assignment solution.

TABLE I
DEFINITION OF NOTATIONS

Notation	Description
A_{max}	The antenna threshold. If an antenna length exceeds A_{max}, an antenna violation happens.
k	The layer number of the 3D grid graph.
$ch(v_i)$	A set of child nodes of v_i. In Fig. 6(a), $ch(v_2)=\{v_3, v_4\}$.
$ch\_e(v_i)$	A set of grid edges connecting v_i to its child nodes. In Fig. 6(a), $ch\_e(v_2)=\{e_3, e_4\}$.
$pinL(v_i)$	If a pin is located at a corresponding node of v_i, $pinL(v_i)$ denotes the layer of the corresponding node. In Fig. 6(b), $pinL(v_5)=3$.
$T(v_i)$	A 2D tree rooted at v_i.
$t_{i,z}$	A 3D tree rooted at $v_{i,z}$. The $t_{i,z}$ is an assignment solution of $T(v_i)$. A 3D tree consists of a set of 3D grid edges and a set of required vias for connecting these grid edges and pins. Fig. 6(c) illustrates a 3D tree $t_{0,2}$.
$S(v_{i,z})$	A set of 3D trees rooted at $v_{i,z}$.
$S(v_i)$	A set of assignment solutions of $T(v_i)$.
$via(\Delta)$	Δ denotes a set of layers, and $via(\Delta)$ represents a set of required vias connecting the layers of Δ.

e_4, $e_5\}$, and $P_N^k = \{v_{0,2}, v_{2,1}, v_{3,1}, v_{5,3}\}$. The single net layer assignment then identifies a 3-D tree, as shown in Fig. 6(c), $E_N^k = \{e_{1,2}, e_{2,1}, e_{3,1}, e_{4,2}, e_{5,3}\}$. AS-SLA is developed to solve this single-net layer assignment problem and avoid antenna violations. Some used notations are introduced in Table I.

1) *Overview of AS-SLA:* AS-SLA is a two-phase algorithm based on a dynamic-programming method. Fig. 7 presents the pseudo code of AS-SLA. Lines 2–13 are the bottom-up phase; and the procedure InitSol initializes a 3-D tree rooted at the leaf node. The procedure EnumSol enumerates all antenna-safe 3-D trees rooted at the internal node $v_{i,z}$ and assigns some wire segments as separators for each 3-D tree while $R(t_{i,z})$ denotes the set of the separators of $t_{i,z}$. For instance, Fig. 8 shows a 3-D tree $t_{i,z}$ that is a sub-tree of $t_{0,0}$, in which $R(t_{i,z})$ contains $\{e_{2,3}, e_{5,4}, e_{7,5}\}$. Notably, a separator, states $e_{a,al}$, in $R(t_{i,z})$ should satisfy the following properties.

Property 1: The layer of a separator must be higher than the layer of its neighboring antennas' grid edges. For instance, in Fig. 8, $\{e_{6,2}, e_{4,2}\}$ and $\{e_{9,4}\}$ are the neighboring antennas of the separator $e_{7,5}$. Therefore, the layer of $e_{7,5}$ is higher than that of $e_{6,2}$, $e_{4,2}$, and $e_{9,4}$.

Property 2: A separator $e_{b,bl}$ appears before $e_{a,al}$ in a path from a leaf to the root, the layer of $e_{a,al}$ must not be higher than $e_{b,bl}$, owing to that if the layer of $e_{a,al}$ is higher than $e_{b,bl}$, $e_{b,bl}$ cannot be regarded as a separator according to the separator's definition.

Algorithm *AC-SLA*
Input: net $N(V_N, E_N)$, pin set P_N^k , 3D grid graph G^k;
1. //Bottom-Up phase
2. **foreach** node v_i in the order given by a postorder traversal of V_N, v_i is not the root.
3. **foreach** layer z from 1 to k
4. **set** $S(v_{i,z})$ to Φ
5. **if** v_i is a leaf node
6. $S(v_{i,z})$←InitSol($v_{i,z}$, P_N^k)
7. **else**
8. $S(v_{i,z})$←EnumSol($v_{i,z}$, $ch(v_i)$, $ch\_e(v_i)$, P_N^k)
9. **end if**
10. PruneSol($S(v_{i,z})$)
11. **end foreach**
12. $S(v_i)$=$S(v_{i,0})\cup S(v_{i,1})\cup ... \cup S(v_{i,k})$
13. **end foreach**
14. //Top-Down phase
15. x=$pinL(v_0)$
16. $S(v_{0,x})$←EnumSol($v_{0,x}$, $ch(v_0)$, $ch\_e(v_0)$, P_N^k)
17. $t_{0,x}$←Select_solution($S(v_{0,x})$)
18. TopDown_Assignment(N, G^k, $t_{0,x}$)
19. **end**

Fig. 7. Pseudo code of AC-SLA.

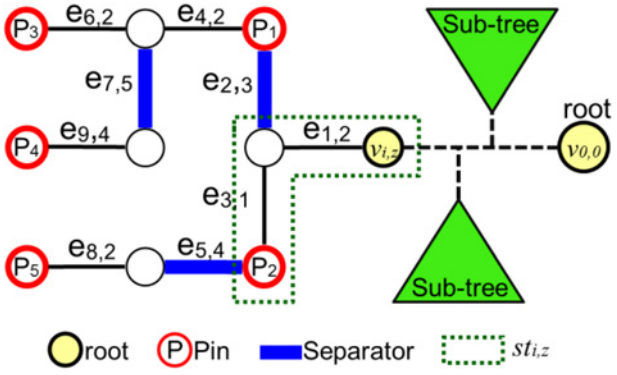


Fig. 8. 3-D sub-tree $t_{i,z}$ with $R(t_{i,z})$ containing $\{e_{2,3}, e_{5,4}, e_{7,5}\}$.

In lines 4–9, the 3-D trees generated by InitSol and EnumSol are inserted into $S(v_{i,z})$. At line 10, the procedure PruneSol discards the inferior layer assignment solutions from $S(v_{i,z})$ to limit the size of $S(v_{i,z})$ in an acceptable range. Next, $S(v_i)$ is obtained by the union of $S(v_{i,z})$, $1\le z\le k$. Lines 15–18 are the top-down phase. Line 16 enumerates all antenna-safe assignment solutions for the entire 3-D tree into $S(v_{0,x})$. Line 17 extracts the minimum-cost assignment solutions from $S(v_{0,x})$. Finally, each wire segment of N is assigned to the corresponding layers according to the minimum-cost solution. The cost of a 3-D tree $t_{i,z}$ is evaluated as follows:

$$cost(t_{i,z}) = sepCost \times |R(t_{i,z})| + viaCost \times numVia(t_{i,z}) + \sum_{e\in t_{i,z}} congCost(e) \quad (4)$$

where *sepCost* denotes a user defined constant for the cost of a single separator and $|R(t_{i,z})|$ represents the separator number

TABLE II
DEFINITION OF NOTATIONS

Notation	Description
$st_{i,z}$	The maximum sub-tree of $t_{i,z}$ rooted at $v_{i,z}$ and surrounded by separators.
$AL(t_{i,z})$	The antenna length of $st_{i,z}$.
$LS(t_{i,z})$	The lowest layer containing at least one separator in $t_{i,z}$.
$TW(t_{i,z})$	The highest layer containing at least one wire segment in $st_{i,z}$.
$R(t_{i,z})$	The set of separators of $t_{i,z}$.
f_i	The Boolean value implying whether $e_{i,z}$ is the separator.

of $t_{i,z}$. Adjusting the value of *sepCost* can control the number of separators. In Section VI, experimental results indicate the effectiveness of adjusting the value of *sepCost*. Two issues in AS-SLA must be addressed: 1) how to enumerate antenna-safe assignment solutions and 2) how to prune the surplus solutions in order to reduce the size of a solution set with the minimum-cost antenna-safe assignment solution still remaining in the solution set.

2) *Enumerating Antenna-Safe LA Solutions:* To address the first issue, InitSol and EnumSol update the attributes of $AL(t_{i,z})$, $LS(t_{i,z})$ and $TW(t_{i,z})$ for each 3-D tree $t_{i,z}$. These attributes are used to judge whether $t_{i,z}$ is an antenna-safe tree in the procedure of EnumSol. Table II introduces the notations used in the procedure of EnumSol. In Fig. 8, $st_{i,z}=\{e_{1,2}, e_{3,1}\}$, $AL(t_{i,z})=2$, $TW(t_{i,z})=2$ and $LS(t_{i,z})=3$. Notably, if $t_{i,z}$ contains no separator, $LS(t_{i,z})$ is initialized to an extremely large constant. If $st_{i,z}$ contains no wire segment, $TW(t_{i,z})$ and $AL(t_{i,z})$ are initialized to zero. Based on the attributes of *AL*, *LS* and *TW*, an antenna-safe 3-D tree can be defined as follows.

Definition 2: Antenna-safe 3-D tree: The $t_{i,z}$ is regarded as an antenna-safe 3-D tree if v_i is the leaf node or v_i is the internal node such that the following three conditions are true. Assume that $v_s \in ch(v_i)$ and $t_{s,ls}$ is one of sub-trees of $t_{i,z}$.

1) Each $t_{s,ls}$ is an antenna-safe 3-D tree.
2) $AL(t_{i,z}) \le A_{max}$.
3) For each $t_{s,ls}$, $ls > \mathrm{TW}(t_{s,ls})$ and $ls \le LS(t_{s,ls})$ if $e_{s,ls} \in R(t_{i,z})$.
4) For each $t_{s,ls}$, $ls < LS(t_{s,ls})$ if $e_{s,ls} \notin R(t_{i,z})$.

The first condition ensures that all sub-trees of $t_{i,z}$ are antenna-safe 3-D trees. The second condition ensures that the antenna length of $st_{i,z}$ is shorter than or equal to A_{max} to avoid antenna violations. The third and fourth conditions let the separators of $t_{i,z}$ conform to Properties 1 and 2, in which wire segment $e_{s,ls}$ connects the root of $t_{s,ls}$ to its adjacent upstream node.

Fig. 9 shows the pseudo code of InitSol. Line 4 calculates the cost of $t_{i,z}$, where $|I|$ denotes the number of required vias connecting the pin's layer to layer z. Lines 5–7 initialize the attribute values of $AL(t_{i,z})$, $LS(t_{i,z})$, and $TW(t_{i,z})$. Fig. 10 shows the pseudo code of EnumSol. For an easy explanation, assume that v_i has three child nodes in the pseudo code. The loop from lines 1–12 enumerates all combinations of the sub-trees of $t_{i,z}$. The $t_{a,la}$, $t_{b,lb}$ and $t_{c,lc}$, are the sub-trees of $t_{i,z}$, and root at $v_{a,la}$, $v_{b,lb}$ and $v_{c,lc}$, respectively. The $v_{a,la}$, $v_{b,lb}$ and $v_{c,lc}$

Procedure **InitSol**
Input: node $v_{i,z}$, 3D pins location P_N^k
1. **Via_set** I= via(pinL(v_i), z)
2. **3D_tree** $t_{i,z}=I$
3. **Separator_set** $R(t_{i,z})=\Phi$
4. $cost(t_{i,z})=|I| * viaCost$
5. $AL(t_{i,z})=0$
6. $LS(t_{i,z})=\infty$
7. $TW(t_{i,z})=0$
8. **insert** $t_{i,z}$ into $S(v_{i,z})$
9. **return** $S(v_{i,z})$

Fig. 9. Procedure InitSol of AC-SLA.

Procedure **EnumSol**
Input: node $v_{i,z}$, $ch(v_i)=\{v_a, v_b, v_c\}$, $ch\_e(v_i)=\{e_a, e_b, e_c\}$, 3D pins set P_N^k
1. **foreach** solution $t_{a,la}$, $t_{b,lb}$ and $t_{c,lc}$ of $S(v_a)$, $S(v_b)$ and $S(v_c)$, respectively. Let $e_{a,la}$, $e_{b,lb}$ and $e_{c,lc}$ are the 3D grid edges connecting the roots of $t_{a,la}$, $t_{b,lb}$ and $t_{c,lc}$ to the 3D nodes $v_{i,la}$, $v_{i,lb}$ and $v_{i,lc}$, respectively.
2. **foreach** combination (f_a, f_b, f_c) // $f_{\{a,b,c\}}=\{0,1\}$,
3. **Via_set** I= *via*(*la*, *lb*, *lc*, *pinL*(v_i), z)
4. **3D_tree** $t_{i,z}= t_{a,la}+t_{b,lb}+ t_{c,lc}+e_{a,la}+e_{b,lb}+ e_{c,lc}+I$
5. **Separator_set** $R(t_{i,z})=R(t_{a,la})+R(t_{b,lb})+R(t_{c,lc})+ f_a*e_{a,la}+f_b*e_{b,lb}+f_c*e_{c,lc}$
6. calculateCost($t_{i,z}, f_a, f_b, f_c$)
7. setAttributes($t_{i,z}, f_a, f_b, f_c$)
8. **if** $t_{i,z}$ **is an antenna-safe 3D tree**
9. **insert** $t_{i,z}$ into $S(v_{i,z})$
10. **endif**
11. **end foreach**
12. **end foreach**
13. **return** $S(v_{i,z})$

Fig. 10. Procedure EnumSol of AC-SLA.

are the corresponding nodes of $ch(v_i)$. Let $e_{a,la}$, $e_{b,lb}$ and $e_{c,lc}$ denote the 3-D grid edges connecting $v_{a,la}$, $v_{b,lb}$ and $v_{c,lc}$ to the 3-D nodes $v_{i,la}$, $v_{i,lb}$ and $v_{i,lc}$, respectively. The loop from lines 2–11 enumerates all combinations of (f_a, f_b, f_c). The two loops explore all 3-D trees rooted at $v_{i,z}$ with different separator assignments. At line 4, a 3-D tree $t_{i,z}$ is constructed by composing $t_{a,la}$, $t_{b,lb}$, $t_{c,lc}$, $e_{a,la}$, $e_{b,lb}$, $e_{c,lc}$ and I. The vias in I connect $v_{i,la}$, $v_{i,lb}$, $v_{i,lc}$ and $v_{i,z}$. If a pin is located at a corresponding node of v_i, vias also connect to this node. Fig. 11 illustrates an example of constructing a 3-D tree $t_{i,3}$, which consists of $t_{a,2}$, $t_{b,3}$, $t_{c,4}$, $e_{a,2}$, $e_{b,3}$, $e_{c,4}$, and I. If $(f_a, f_b, f_c)=\{1, 0, 1\}$, $e_{a,2}$ and $e_{c,4}$ are regarded as the separators. Line 5 builds the separator set of $t_{i,z}$. Line 6 calculates the cost of $t_{i,z}$ by (5), and then line 7 sets the attribute values of $AL(t_{i,z})$, $LS(t_{i,z})$, and $TW(t_{i,z})$ by (6)–(8), respectively

$$cost(t_{i,z}) = |I| * viaCost + \sum_{vs \in ch(vi)} \begin{matrix} cost(t_{s,ls}) + f_s * sepCost \\ +congCost(e_{s,ls}) \end{matrix} \tag{5}$$

$$AL(t_{i,z}) = \sum_{vs \in ch(vi)} (1 - f_s) * (AL(t_{s,ls}) + 1) \tag{6}$$

$$LS(t_{i,z}) = \min_{vs \in ch(vi)} (LS(t_{s,ls}), (1 - f_s) * k + ls) \tag{7}$$

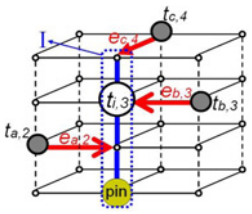


Fig. 11. Illustrative example for constructing a 3-D tree $t_{i,3}$, which consists of $t_{a,2}$, $t_{b,3}$, $t_{c,4}$, $e_{a,2}$, $e_{b,3}$, $e_{c,4}$, and I.

$$TW(t_{i,z}) = \max_{vs \in ch(vi)} ((1 - f_s) * TW(t_{s,ls})). \tag{8}$$

Line 8 verifies whet her $t_{i,z}$ is antenna-safe. Only the antenna-safe 3-D tree can be inserted into the solution set $S(v_{i,z})$.

3) *Pruning Inferior Assignment:* To limit the size of the solution set in an acceptable range, PruneSol discards the inferior Assignment from $S(v_{i,z})$. The flexibility of an assignment solution and inferior assignment are defined as follows.

Definition 3: Flexibility: Given a net N, assume $t_{0,0}$ is an antenna-safe 3-D tree of N, and $t_{i,z}$ is one of its antenna-safe 3-D sub-trees. Let $e_{a,al}$ be a 3-D edge of $t_{0,0}$ in the path from the root of $t_{i,z}$ to the root of $t_{0,0}$, and al is the layer of $e_{a,al}$. If $e_{a,al}$ is a separator, the inequality, $TW(t_{i,z}) < al \leq LS(t_{i,z})$, must hold because Properties 1 and 2 are violated when $al \leq TW(t_{i,z})$ and $\mathrm{al} > \mathrm{LS}(t_{i,z})$, respectively. On the contrary, if $e_{a,al}$ is not a separator, the inequality, $\mathrm{al} < \mathrm{LS}(t_{i,z})$, must hold because Property 1 is violated as $al \geq LS(t_{i,z})$. Thus, the large layer range between $TW(t_{i,z})$ and $LS(t_{i,z})$ implies that more layers are the assignment candidates for $e_{a,al}$. In this paper, if the layer range of a 3-D tree $t^w_{i,z}$ is a sub-range of that of another 3-D tree $t^u_{i,z}$, $t^u_{i,z}$ is regarded to have better flexibility than $t^w_{i,z}$.

Definition 4: Inferior assignment: $t^w_{i,z}$ and $t^u_{i,z}$ are the 3-D trees in $S(v_{i,z})$, each of which is a layer assignment solution for $T(v_i)$. 3-D tree $t^w_{i,z}$ is regarded as an inferior assignment if $t^u_{i,z}$ holding the following equations exists:

$$\begin{aligned} &AL(t^w_{i,z}) \geq AL(t^u_{i,z}), cost(t^w_{i,z}) \geq cost(t^u_{i,z}) \\ &TW(t^w_{i,z}) \geq TW(t^u_{i,z}) \quad \text{and} \quad LS(t^w_{i,z}) \leq LS(t^u_{i,z}) \end{aligned} \tag{9}$$

where the layer range of $t^w_{i,z}$ is totally covered by that of $t^u_{i,z}$, so $t^w_{i,z}$ has worse flexibility than $t^u_{i,z}$. Moreover, $t^w_{i,z}$ has a higher cost and longer antenna than $t^u_{i,z}$, implying $t^w_{i,z}$ is totally worse than $t^u_{i,z}$. Accordingly, $t^w_{i,z}$ should be discarded from $S(v_{i,z})$.

To explain that pruning inferior assignment $t^w_{i,z}$ would not degrade the solution quality of AS-SLA, we first claim that a 3-D tree containing $t^w_{i,z}$ must be an inferior assignment, which ensures that the solution of AS-SLA containing $t^w_{i,z}$ is not of minimum cost. Assume a 3-D tree $t^w_{r,x}$ consists of $\{t^w_{i,z}, t_{j,h}, e_{i,z}, e_{j,h}\}$ where $e_{i,z}$ and $e_{j,h}$ connect the roots of $t^w_{i,z}$and $t_{j,h}$ to the root of $t^w_{r,x}$, respectively. If $t^w_{i,z}$ is inferior to $t^u_{i,z}$, we can guarantee that $t^w_{r,x}$ must be inferior to $t^u_{r,x}$ that consists of $\{t^u_{i,z}, t_{j,h}, e_{i,z}, e_{j,h}\}$ since the attributes of $t^w_{r,x}$ and $t^u_{r,x}$ satisfy (9). The attributes of $t^w_{r,x}$ ($t^u_{r,x}$) are computed by (5)–(8) considering the separator assignments of $e_{i,z}$ and $e_{j,h}$ and the attributes of $t_{j,h}$ and $t^w_{i,z}$ ($t^u_{i,z}$). According to (5)–(8), if the attributes of $t^w_{i,z}$ and $t^u_{i,z}$ satisfy (9), the attributes of $t^w_{r,x}$ and $t^u_{i,z}$ must satisfy (9) for any $t_{j,h}$ and any separator assignments of $e_{i,z}$ and $e_{j,h}$. Thus, any 3-D tree containing $t^w_{i,z}$ must be an inferior assignment.

To identify the inferior assignments in $S(v_{i,z})$, the most intuitive method is to compare all pairs of the solutions in $S(v_{i,z})$. As the original size of $S(v_{i,z})$ is n, the time complexity of the intuitive method is $O(n^2)$. The method of complexity $O(n^2)$ is unrealistic as n increases up to a very large number. Therefore, in the next section, a dynamic-programming-based inferior assignment pruning algorithm (DPIAP) is proposed, which can prune all inferior assignments and the time complexity of DPIAP is $O(\mathrm{n} + \mathrm{k2Amax})$. As n is commonly larger than $k^2 A_{max}$, $O(\mathrm{n} + \mathrm{k2Amax})$ is faster than $O(n^2)$.

B. Acceleration Techniques for AS-SLA

Before detailing the proposed acceleration techniques, we first analyze the time complexity of AS-SLA to show the most time consuming part in AS-SLA. Assuming the maximum size of $S(v_{i,z})$ is m after pruning inferior assignments. Also, the maximum size of $S(v_i)$ is km since $S(v_i)$ is the union of all $S(v_{i,z})$ for $1 \leq z \leq k$. Based on this assumption, the time complexity of each procedure in AS-SLA is analyzed as follows.

Lemma 1: The time complexity of EnumSol for $v_{i,z}$ is O $((2\,km)^q)$ as the child number of v_i is q.

Proof: Fig. 10 shows the pseudo code to enumerate all 3-D trees rooted at $v_{i,z}$ as q is 3. The loop from lines 1–12 enumerates all combinations of the child node's solutions. As the child number of v_i is q and the maximum number of each child tree's assignment solutions is km, the loop from lines 1–11 runs at most $(km)^q$ times to enumerate $(km)^q$ combinations of the child node's solutions. Moreover, the inner-loop from lines 2 to 10 runs 2^q times, so the time complexity of EnumSol is $O((km)^q) \times O(2^q) = O((2\,km)^q)$ and the amount of the enumerated solution space is $(2\,km)^q$. ■

Lemma 2: With the intuitive pruning method, the time complexity of PruneSol for $\mathrm{S}(v_{i,z})$ is $(2\,km)^{2q}$.

Proof: By *Lemma* 1 $S(v_{i,z})$ contains at most $(2\,km)^q$ solutions, the time complexity of comparing all pairs of the solutions in $S(v_{i,z})$ is $O((2\,km)^q) \times O((2\,km)^q) = O((2\,km)^{2q})$. ■

Theorem 1: Given a net $N(V_N, E_N)$, the worst time complexity of AS-SLA to assign N is $O(k(km)^6|E_N|)$ as PruneSol adopts the intuitive pruning method, where $|E_N|$ denotes the number of edges in E_N.

Proof: In the bottom-up phase of AS-SLA, there are an inner loop from lines 3–11 and an outer loop from lines 2–13. Routine InitSol takes constant time to complete the operation and, by *Lemmas* 1 and 2, PruneSol is the most time consuming part in the inner loop. As the time complexity of PruneSol is $O((2\,km)^{2q})$ and the number of iterations for inner loop is k, the time complexity of the inner loop is $O(k(2\,km)^{2q})$. Furthermore, the number of iterations for outer loop is $|E_N|$. Thus, the time complexity of the outer loop is $O(k(2\,km)^{2q}|E_N|)$. As the number of child nodes of each node except the root in V_N is at most three ($q \leq 3$), the worst time complexity of the bottom-up phase is $O(k(km)^6|E_N|)$. In the top-down phase, since the child number of the root is at most four, the worst time complexity of EnumSol is $O((2\,km)^4)$ at line 16. Also, the time complexity of Select_solution to pick the minimum-cost

solution is $O((2\,km)^4)$. Finally, the time complexity of TopDown_Assignment to assign each wire segment to its corresponding layer is $O(|E_N|)$. The time complexity of this phase is $O((2\,km)^4)+O((2\,km)^4)+O(|E_N|)=O((km)^4+|E_N|)$ that is smaller than $O(k(km)^6|E_N|)$. Thus, the time complexity of AS-SLA is $O(k(km)^6|E_N|)$. ■

1) *DP-Based Inferior Assignment Pruning (DPIAP):* Theorem 1 indicates that PruneSol is the most time consuming part in AS-SLA. Thus, we present a dynamic-programming-based inferior assignment pruning algorithm (DPIAP) to reduce the time complexity of PruneSol from $O(n^2)$ to $O(\mathrm{n}+\mathrm{k2Amax})$ where n denotes the size of $S(v_{i,z})$, i.e., maximally up to $(2\,km)^q$ revealed by *Lemma* 2.

DPIAP consists of the intrabin comparison stage and the interbin comparison stage, where each solution in $S(v_{i,z})$ is categorized into one of a set of bins according to its attributes of TW, LS and AL. For instance, a solution $t_{i,z}$ with $TW(t_{i,z})=3$, $LS(t_{i,z})=5$, and $AL(t_{i,z})=15$ is assigned to the bin $B[3, 5, 15]$. At first, the intrabin comparison stage preserves only the minimum-cost solution and discards the other solutions in each bin because the other solutions in the same bin are totally worse than the minimum-cost one. The intrabin comparison stage can significantly reduce solution set size. However, this stage cannot prune all inferior assignment solutions since the solutions in different bins are not compared with one another.

After the intrabin comparison stage, the remaining solution in bin $B[\gamma, \delta, \varepsilon]$ is denoted by $s[\gamma, \delta, \varepsilon]$, in which γ, δ and ε are the values of TW, LS and AL of the remaining solution, respectively. If $B[\gamma, \delta, \varepsilon]$ is empty, a pseudo solution $s[\gamma, \delta, \varepsilon]$ is created and the cost of the pseudo solution is set to be a very large constant. We define two relations between any two of the remaining solutions. First, solution $s_i[\gamma_i, \delta_i, \varepsilon_i]$ is said to cover another solution $s_j[\gamma_j, \delta_j, \varepsilon_j]$ if one of the following conditions holds.

1) $\gamma_i<\gamma_j, \delta_i\geq\delta_j, \varepsilon_i=\varepsilon_j$
2) $\gamma_i\leq\gamma_j, \delta_i>\delta_j, \varepsilon_i\leq\varepsilon_j$
3) $\gamma_i\leq\gamma_j, \delta_i\geq\delta_j, \varepsilon_i=\varepsilon_2$

Secondly, solution $s_i[\gamma_i, \delta_i, \varepsilon_i]$ is said to critically cover another solution $s_j[\gamma_j, \delta_j, \varepsilon_j]$ if one of the following conditions holds.

1) $\gamma_i=\gamma_j\text{-}1,\ \delta_i=\delta_j,\ \varepsilon_i=\varepsilon_j$
2) $\gamma_i=\gamma_j,\ \delta_i=\delta\mathrm{j}+1,\ \varepsilon_i=\varepsilon_j$
3) $\gamma_i=\gamma_j,\ \delta_i=\delta_j,\ \varepsilon_i=\varepsilon_j\text{-}1$

To express the covering relations among the remaining solutions, a directed covering graph is built, in which each node denotes an assignment solution, and a directed edge from $s_i[\gamma_i, \delta_i, \varepsilon_i]$ to $s_j[\gamma_j, \delta_j, \varepsilon_j]$ denotes that $s_i[\gamma_i, \delta_i, \varepsilon_i]$ critically covers $s_j[\gamma_j, \delta_j, \varepsilon_j]$. For example, Fig. 12 shows a directed covering graph for a design with four layers and A_{max} of 3. Assignments $s_5[1, 3, 1]$, $s_{11}[2, 4, 1]$ and $s_{13}[2, 3, 0]$ critically cover $s_{14}[2, 3, 1]$ and thus they are the parents of s_{14} [2, 3, 1] in the graph. Moreover, if $s_i[\gamma_i, \delta_i, \varepsilon_i]$ covers $s_j[\gamma_j, \delta_j, \varepsilon_j]$, $s_i[\gamma_i, \delta_i, \varepsilon_i]$ is an ancestor of $\mathrm{s}_j[\gamma_j, \delta_j, \varepsilon_j]$ in the graph. A lemma highlights a property of a directed covering graph.

Lemma 3: $\mathrm{s_i}[\gamma_i, \delta_i, \varepsilon_i]$ is an inferior assignment solution if the minimum cost of its ancestors, named as minimum ancestor cost (mac), in the associated directed covering graph is less than the cost of $\mathrm{s_i}[\gamma_i, \delta_i, \varepsilon_i]$; otherwise, $\mathrm{s_i}[\gamma_i, \delta_i, \varepsilon_i]$ is non-inferior assignment.

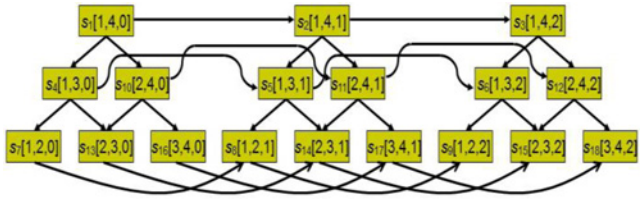


Fig. 12. Directed covering graph for a design with four layers and that Amax is 3.

Procedure Inter-bin comparison
Input: the remaining solutions in $S(v_{i,z})$ after intra-bin comparison, layer number k, antenna length limitation A_{max}
1. **Create** a solution array $s[k\times k\times A_{max}]$ ← the remaining solutions in $S(v_{i,z})$ + pseudo solutions
2. **Build** a directed covering graph
3. **for**(int γ=1; $\gamma\leq k$; γ++)
4. **for**(int δ=k; $\delta>\gamma$; δ--)
5. **for**(int ε=0; $\varepsilon<A_{max}$; ε++)
6. s_i denotes $s[\gamma, \delta, \varepsilon]$
7. **foreach** parent solution, denoted s_j, of s_i
8. $mac(s_i)$=**min**($mac(s_i)$, $mac(s_j)$, $cost(s_j)$)
9. **end foreach**
10. **if** $mac(s_i) > cost(s_i)$
11. **Label** s_i as an inferior assignment
12. **end if**
13. **end for**
14. **end for**
15. **end for**
16. **Update** $S(v_{i,z})$ to prune the inferior assignments
17. **return** $S(v_{i,z})$

Fig. 13. Pseudo code of the interbin comparison stage.

Proof: By Definition 4, $s_i[\gamma_i, \delta_i, \varepsilon_i]$ can be regarded as an inferior assignment solution if another solution not only has better attributes of TW, LS and AL than $s_i[\gamma_i, \delta_i, \varepsilon_i]$, but also has smaller cost than $s_j[\gamma_j, \delta_j, \varepsilon_j]$. Based on the definition of the directed covering graph, the ancestors of $s_i[\gamma_i, \delta_i, \varepsilon_i]$ must have better attributes of TW, LS and AL than $s_i[\gamma_i, \delta_i, \varepsilon_i]$. If any ancestor of $s_i[\gamma_i, \delta_i, \varepsilon_i]$ has less cost than $s_i[\gamma_i, \delta_i, \varepsilon_i]$, $s_i[\gamma_i, \delta_i, \varepsilon_i]$ is an inferior assignment. Namely, if the cost of $s_i[\gamma_i, \delta_i, \varepsilon_i]$ is smaller than its minimum-cost ancestor, $s_i[\gamma_i, \delta_i, \varepsilon_i]$ is non-inferior assignment. ■

For example, in Fig. 12, the ancestors of $s_{11}[2, 4, 1]$ are $s_1[1, 4, 0]$, $s_2[1, 4, 1]$ and $s_{10}[2, 4, 0]$. If the costs of $s_{11}[2, 4, 1]$, $s_1[1,4, 0]$, $s_2[1, 4, 1]$ and $s_{10}[2, 4, 0]$ are respectively 65, 52, 60 and 75, $s_{11}[2, 4, 1]$ is an inferior assignment since the *mac* of $s_{11}[2, 4, 1]$ is 52 and smaller than its own cost.

By *Lemma* 3, in the interbin comparison stage, we need to compute the *mac* for each remaining assignment, and then distinguish whether each assignment can be discarded. Fig. 13 shows the pseudo code of the interbin comparison stage, in which $mac(s_i)$ and $cost(s_i)$ denotes the *mac* and cost of assignment solution s_i, respectively. At line 1, a solution array is created to store the remaining solutions and pseudo solutions; the size of the solution array is $\mathrm{k}\times\mathrm{k}\times\mathrm{Amax}$. Initially, for each assignment solution s_i, $mac(s_i)$ is initialized

to a very large value; the subsequent process will identify the actual value of $mac(s_i)$. Next, a directed covering graph is built at line 2 to express the covering relations among the solutions in the solution array. The nested loops from lines 3 to 15 explore each solution from the root toward leaves. In the case of Fig. 12, the index of each assignment solution denotes its explored order. Lines 7–9 identify the actual *mac* for the current explored solution s_i, and lines 10–11 examine whether s_i is an inferior assignment. Finally, the solution set $S(v_{i,z})$ is updated to prune all inferior assignments. The time complexity of DPIAP is analyzed as follows.

Lemma 4: For each 3-D node $v_{i,z}$, i≠0, after pruning inferior assignments, the number of remaining solutions in $S(v_{i,z})$ is at most $(k(k–1)A_{max})/2$.

Proof: A k-layer design has $k(k–1)/2$ layer ranges and the antenna length for each assignment solution ranges from 0 to $A_{max}–1$, so the intrabin comparison stage creates $(k(k–1)/2) \times A_{max}$ bins to accommodate the solutions of different layer ranges and antenna lengths. As each bin preserves at most one solution after intrabin comparison stage, the total number of remaining solution is at most $(k(k–1)A_{max})/2$. ■

Lemma 5: For an $S(v_{i,z})$ of size n, the time complexity of DPIAP is $O(n + k2Amax)$.

Proof: In the intrabin comparison stage of DPIAP, the time complexities of assigning each solution to its corresponding bin and selecting the minimum-cost solution in each bin are both $O(n)$ because each solution must be scanned exactly once. Finally, the time complexity of collecting the remaining solution from each bin to rebuild $S(v_{i,z})$ is $O(k^2A_{max})$ since there are $(k(k–1)A_{max})/2$ bins. Thus, the time complexity of the intrabin comparison stage is $O(n) + O(k^2A_{max}) = O(\mathrm{n} + \mathrm{k2Amax})$. In the interbin comparison stage shown in Fig. 13, the time complexity of creating and initializing solution array at line 1 is $O(k^2A_{max})$. As there are $(k(k–1)A_{max})/2)$ solutions in the solution array and each solution has at most three parents, the time complexity of building the directed covering graph at line 2 is $O((k(k–1)A_{max})/2) \times 3) = O(k^2A_{max})$. Next, the nested loops from lines 3 to 15 altogether run $(k(k–1)A_{max})/2)$ iterations to explore every solution; the inner loop from lines 7 to 9 runs at most three iterations because a solution has at most three parents. Accordingly, the time complexity of the operations during lines 3–15 is $O(3 \times (k(k–1)A_{max})/2) = O(k^2A_{max})$. The time complexity to updating solution set $S(v_{i,z})$ at line 16 is $O(k^2A_{max})$. Totally, the time complexity of the interbin comparison stage is $O(k^2A_{max})$. Due to the time complexities of $O(\mathrm{n} + \mathrm{k2Amax})$ and $O(k^2A_{max})$ respectively for the intrabin and interbin comparison stages, the time complexity of DPIAP is $O(\mathrm{n} + \mathrm{k2Amax}) + O(k^2A_{max}) = O(\mathrm{n} + \mathrm{k2Amax})$. ■

Theorem 2: Given a net $N(V_N, E_N)$, the worst time complexity of AS-SLA to assign N is $O(k(km)3 + k2Amax)|E_N| + (km)^4)$ as PruneSol *adopts DPIAP, where $|E_N|$ represents the number of edges in $|E_N|$.*

Proof: As the maximum number of the solutions enumerated by EnumSol is $(2\,km)^q$, the time complexity of PruneSol adopting DPIAP is $O((2\,km)\mathrm{q} + \mathrm{k2Amax})$. PruneSol is still the bottleneck in the bottom-up phase, so the worst time complexity of the bottom-up phase is $O(k(km)3 + \mathrm{k2Amax})|E_N|)$. The worst time complexity of the top-down phase is still $O((km)^4 + |E_N|)$. Moreover, the worst time complexity of entire AS-SLA is $O(k(km)3 + \mathrm{k2Amax})|E_N|) + O((km)^4 + |E_N|) = O((k(km)3 + \mathrm{k2Amax})|E_N| + (km)^4)$. ■

Corollary 1: AS-SLA adopting DPIAP in PruneSol has less time complexity than AS-SLA adopting the intuitive pruning method in PruneSol.

Proof: The time complexities of AS-SLA as PruneSol adopts DPIAP and intuitive pruning method are respectively $O(k(km)3 + \mathrm{k2Amax})|E_N| + (km)^4)$ and $O(k(km)^6|E_N|)$, where m is the maximum size of $S(v_{i,z})$ after pruning inferior assignment solutions. *Lemma* 4 implies that the number of remaining solutions in $S(v_{i,z})$ is at most $(k(k–1)A_{max})/2$ after pruning inferior assignment solutions, so m can be regarded as $O(k^2A_{max})$. Accordingly, the worst time complexities of AS-SLA as PruneSol adopts DPIAP and the intuitive pruning method are $O(k^{10}A^3_{max}|E_N| + k^{12}A^4_{max})$ and $O(k^{13}A^6_{max}|E_N|)$, respectively. ■

Note that, k and A_{max} are not big constants, so the runtime of AS-SLA is acceptable.

2) *Heuristic Solution Pruning Scheme:* Pruning the inferior assignment solutions can reduce the solution space; however, the execution time of the subsequent process will be long if the remaining solution space is still considerable. In this section, a heuristic scheme is proposed and integrated into DPIAP to limit the remaining solution space. The original solution pruning scheme using (9) only discards the inferior assignments. In contrast, the heuristic scheme prunes the inferior assignments as well as the bad-quality non-inferior assignments. *Lemma* 4 reveals that the number of remaining solutions is at most $(k(k–1)A_{max})/2$. The proposed heuristic pruning scheme can reduce the maximum number of remaining solutions to $(k(k–1)\lfloor A_{max}/\tau \rfloor)/2$, where τ is a user defined positive integer not exceeding A_{max}.

The heuristic pruning scheme prunes an assignment solution $t^w_{i,z}$ if the following equations hold for a certain assignment solution $t^u_{i,z}$:

$$\begin{aligned} &AL(t^w_{i,z})/\tau \geq AL(t^u_{i,z})/\tau, cost(t^w_{i,z}) \geq cost(t^u_{i,z}) \\ &TW(t^w_{i,z}) \geq TW(t^u_{i,z}) \quad \text{and} \quad LS(t^w_{i,z}) \leq LS(t^u_{i,z}). \end{aligned} \tag{10}$$

The concept of the heuristic pruning scheme is to prune $t^w_{i,z}$ if its antenna length is similar to that of $t^u_{i,z}$ but its cost and flexibility are both worse than $t^u_{i,z}$. For instance, assume that [*AL*, *cost*, *TW*, *LS*] of $t^w_{i,z}$ and $t^u_{i,z}$ are [50, 300, 3, 4] and [51, 150, 1, 6], respectively. Obviously the cost and flexibility of $t^w_{i,z}$ are both worse than $t^u_{i,z}$ but the antenna length of $t^w_{i,z}$ is shorter than $t^u_{i,z}$ by one, which does not satisfy (9). However, (10) holds as τ is set to be two, and thus the heuristic pruning scheme will discard $t^w_{i,z}$. The heuristic pruning scheme reduces the solution set size and maintains every antenna length level in the solution set.

The heuristic pruning scheme can be easily integrated into DPIAP by simply replacing the term $AL(t_{i,z})$ used in original DPIAP with $\lfloor AL(t_{i,z})/\tau \rfloor$. DPIAP integrated with the heuristic pruning scheme is denoted by heuristic-DPIAP whose time complexity for $S(v_{i,z})$ is $O(\mathrm{n} + \mathrm{k2Amax}/\tau)$, where n is the size of $S(v_{i,z})$, and the maximum size of $S(v_{i,z})$ after heuristic-DPIAP is $(k(k–1)\lfloor A_{max}/\tau \rfloor)/2$.

Heuristic-DPIAP can be adopted to replace the original solution pruning method (line 10, Fig. 7) to speed up AS-SLA. As a result, the time complexity of AS-SLA with heuristic-DPIAP is $O(k^{10}A_{max}^3|E_N|/\tau 3 + \text{k12Amax4}/\tau^4)$. Notably, AS-SLA with heuristic-DPIAP cannot guarantee to obtain the minimum-cost antenna-safe solution because the minimum-cost solution may be pruned by heuristic-DPIAP. By experimental results, when τ is set between 2 to 10, AS-SLA has good acceleration and can obtain a satisfactory assignment result.

V. Proposed Antenna Avoidance Layer Assignment Flow (AALA)

The flow of AALA consists of three stages, i.e., initial layer assignment, wire overflow reduction and post optimization, which resemble that in [19]. AALA and the method in [19] both adopt a negotiation-based assignment scheme, whose essential idea is to allow the occurrence of wire congestion violations in the initial assignment, and then to gradually reduce the violations by penalizing more heavily and reassigning the nets with wire congestion violations. When the negotiation-based assignment scheme is used, the early assigned nets and latter assigned nets can fairly compete for the routing resource to yield better layer assignment results. The experiments in [19] reveal that the impact of net ordering on the assignment quality becomes slight if the negotiation-based assignment scheme is adopted. Accordingly, the layer assignment order of nets simply follows the increasing order of their net number that has no special meaning in the benchmarks used in this paper.

In the initial layer assignment stage, the minimal via assignment solution of each net is greedily identified by NANA without considering the antenna rule and wire congestion constraints. Next, in the wire overflow reduction stage, the congestion cost is formulated for each grid edge. The nets with overflows or antenna violations are sequentially ripped up and reassigned. For each net that is assigned to violate wire congestion constraints, the net will be ripped up and reassigned again in the next round. At the end of each round, the congestion cost of each overflowed grid edge increases to prevent the wire segments from overusing the overflowed grid edges in the next round. The wire overflow reduction stage repeats iteratively until the assignment result of each net conforms to wire congestion constraints or the overflows do not decline for three consecutive rounds. In wire overflow reduction stage, NANA is used to assign the short nets; meanwhile, the long nets are assigned using AS-SLA. Notably, a net is regarded as a short net if the length of the net is less than A_{max}; otherwise, the net is regarded as a long net. As the assignment result of a short net must conform to the antenna rule, performing NANA to assign a short net can always identify the minimum-cost antenna-safe solution, which is also more efficient than performing AS-SLA.

Finally, in the post optimization stage, NANA and AS-SLA are respectively used to rip-up and reassign each short net and long net to further reduce the via count. If AS-SLA identifies an antenna-safe solution with wire congestion violations for a net, this net will be reassigned again by NANA to meet wire congestion constraints even the number of antenna-violations increases. Thus, the final assignment result must conform to wire congestion constraints.

TABLE III
ISPD'08 Benchmarks

Benchmark	Grids	Layers	A_{max}	Long Nets	Short Nets
adaptec1	324×324	6	85	11300	165415
adaptec2	424×424	6	85	7818	200154
adaptec3	774×779	6	100	26804	341690
adaptec4	774×779	6	100	22609	378451
adaptec5	465×468	6	60	51191	496882
newblue1	399×399	6	100	3372	267341
newblue2	557×463	6	60	17224	356566
newblue3	973×1256	6	75	19079	422926
newblue4	455×458	6	75	20973	510319
newblue5	637×640	6	75	50412	841508
newblue6	463×464	6	50	49586	785681
newblue7	488×490	8	37	137863	1509547
bigblue1	227×227	6	60	18166	178719
bigblue2	468×471	6	75	11657	417311
bigblue3	555×557	8	60	31221	634408
bigblue4	403×405	8	37	83314	1050221

In each stage, AS-SLA and NANA are both designed based on the objective function (4) to perform layer assignment, but NANA always treats $|R(t_{i,z})|$ as zero because separators are not used in NANA. Although the objective function (4) is adopted by all stages of AALA, the formula of $congCost(e)$ in (4) will change in different stages to achieve different purposes. The congestion cost is fixed as zero in the initial layer assignment stage that NANA identifies the minimal via count solution for each net. During the wire overflow reduction stage, if grid edge e frequently overflows, the congestion cost of e gradually increases and is formulated as follows:

$$congCost(e) = p_e * (1 + (h_e)^2) \tag{11}$$

where p_e and h_e denote the congestion penalty and the history cost. The congestion penalty is inspired by the idea in [10], and defined as follows:

$$p_e = 1 + \frac{\alpha}{1 + \exp^{\beta*(cap(e)-dem(e))}} \tag{12}$$

where $cap(e)$ and $dem(e)$ refer to the capacity and demand of e, respectively. The congestion penalty increases dramatically as demand approximates capacity, which actively attracts demand away from over-capacity edges and toward within-capacity ones to meet wire congestion constraints in AS-SLA and NANA. If demand is largely below or above capacity, the congestion penalty increases mildly. Notably, α and β are user defined constants; α is set to ten and β is set to 0.3 in this paper. At the end of a round in the wire overflow reduction stage, the history cost h_e increases by one as the grid edge e overflows in this round; otherwise, h_e remains unchanged if e does not overflow. The value of h_e, in the first round is initialized to zero, and, in the k-th round, can be expressed as

$$h_e^{k+1} = \begin{cases} h_e^k + 1 & \text{if } e \text{ is overflowed} \\ h_e^k & \text{otherwise.} \end{cases} \tag{13}$$

TABLE IV
COMPARISON BETWEEN THE PROPOSED ANTENNA AVOIDANCE LAYER ASSIGNMENT AND PREVIOUS WORKS

Benchmark	COLA [17]			NVM [19]			LAVA [2]			AALA		
	#vn	vias (10^5)	cpu* (min)	#vn	vias (10^5)	cpu (min)	#vn	vias (10^5)	cpu* (min)	#vn	vias (10^5)	cpu (min)
adaptec1	911	17.69	0.46	709	16.69	0.80	602	17.51	0.73	4	16.72	14.82
adaptec2	879	19.30	0.43	712	18.31	0.70	568	19.07	0.64	0	18.33	12.86
adaptec3	2959	34.91	1.38	2919	32.90	2.11	2194	34.58	1.94	5	33.00	60.03
adaptec4	2009	32.15	1.25	1925	30.82	1.76	1931	31.93	1.61	4	30.90	41.11
adaptec5	4166	52.40	1.65	3744	49.30	2.28	2465	51.90	2.09	0	49.43	53.53
newblue1	328	22.22	0.38	460	21.42	0.58	273	24.95	0.53	6	21.43	4.93
newblue2	681	29.46	0.66	534	28.14	0.94	444	29.15	0.86	1	28.18	11.79
newblue3	466	30.23	0.99	429	29.00	1.58	251	29.42	1.44	1	29.08	29.48
newblue4	874	47.05	1.33	849	44.73	1.68	617	46.59	1.54	0	44.77	20.23
newblue5	3009	84.51	2.25	2766	80.16	3.52	2137	83.79	3.23	0	80.30	77.93
newblue6	3453	74.66	1.76	3280	71.01	2.39	2736	73.83	2.19	0	71.12	33.27
newblue7	10286	166.01	5.10	8628	157.21	6.47	5844	164.52	5.93	0	157.50	354.41
bigblue1	1841	18.73	0.64	1459	17.60	0.93	1423	18.57	0.85	0	17.65	19.71
bigblue2	392	42.11	0.87	389	40.32	1.24	264	41.72	1.13	0	40.34	16.25
bigblue3	3576	52.43	1.51	3631	50.55	2.49	2692	51.99	2.28	0	50.66	233.01
bigblue4	7676	109.14	2.78	8627	104.69	4.32	5230	108.28	3.96	0	104.93	301.91
Sum	43506			41261			29671			21		
Ratio		1.049	0.036		0.998	0.053		1.046	0.123		1	1

*AMD Dual Core Opteron Processor 2.2-GHz CPU.

Finally, the following congestion cost formula is used in the post optimization stage to largely penalize the congestion situation to avoid overflows:

$$congCost(e)=\begin{cases}\sigma * (1 + dem(e) - cap(e)), if dem(e) \geq cap(e)\\ 0, otherwise,\end{cases} \tag{14}$$

where σ denotes an extremely large constant. Overflow minimization is regarded as the first objective as NANA and AS-SLA adopt (14) to formulate the congestion cost in the post optimization stage. As a result, the assignment result after the post optimization stage must conform to wire congestion constraints.

VI. EXPERIMENT RESULTS

The proposed algorithms were implemented in C/C++ language on a 2.4 GHz Intel Xeon-based Linux server with 48 GB memory, and use the benchmarks from the ISPD'08 global routing contest [22]. Each benchmark has six or eight routing layers. The preferred direction in even layers is horizontal, while that in odd layers is vertical. A used net edge that does not follow the preferred direction produces a wire overflow. To compare AALA with previous works, each algorithm reads the same the 2-D global routing results of NTHU-Route 2.0 [9] for ISPD'08 benchmarks. Then, each layer assignment algorithm transforms 2-D routing results to 3-D. Table III lists the grid size and layer number of each benchmark [22] in the second and third columns, respectively; the value of A_{max} setting for each benchmark is shown in the fourth column, which is same as the setting in [2]. The fifth column (sixth column) displays the number of nets with the longer (shorter) length than A_{max} in the routing result obtained by NTHU-Route 2.0 for each benchmark.

TABLE V
COMPARISON BETWEEN LAVA AND EXTENDED-LAVA

Bench-mark	$LAVA_O$			$LAVA_E$		
	#vn	vias (10^5)	cpu (min)	#vn	vias (10^5)	cpu* (min)
adaptec1	781	19.01	1.00	629	19.57	1.20
adaptec2	684	21.09	0.99	671	21.86	1.06
adaptec3	2879	38.00	3.14	2450	39.44	3.73
adaptec4	2556	35.55	2.23	1809	36.29	2.62
adaptec5	3194	57.82	3.17	2855	59.28	3.67
newblue1	334	26.97	0.84	283	27.72	0.91
newblue2	585	32.46	1.30	508	33.18	1.40
newblue3	318	33.02	2.14	274	34.29	2.48
newblue4	744	51.66	2.09	680	53.51	2.38
newblue5	2758	90.80	4.49	2411	91.84	5.04
newblue6	3033	78.48	3.38	2636	79.90	4.07
newblue7	7053	182.10	10.59	6003	188.20	11.91
bigblue1	1764	21.35	1.39	1662	21.81	1.56
bigblue2	329	47.94	1.63	314	49.21	1.97
bigblue3	3156	57.01	3.93	2922	58.35	4.59
bigblue4	6208	117.03	7.09	6081	118.59	8.66
Sum	36377			32188		
Ratio		1.00	1.00		1.026	1.152

Many parameters are introduced to some equations in this paper. All parameters are set based on our empirical senses. Only one set of parameter values is used to run through all benchmarks, but not one set of parameter values for one benchmark.

A. Comparison Among AALA and Existing Works

Table IV compares AALA with the recent layer assignment works, in which the *sepCost* and *viaCost* is respectively set to 1 and 100 for AALA. In Table IV, COLA [17] and NVM [19] focus on the via count minimization but does not consider the antenna effect, and LAVA [2] is an antenna avoidance layer assignment algorithm which addresses on eliminating

TABLE VI
EFFECTIVENESS OF PROPOSED ACCELERATION TECHNIQUES

Benchmark	$AALA_1$			$AALA_2$			$AALA_3$ (τ=2)			$AALA_3$ (τ=6)			$AALA_3$ (τ=10)		
	#vn	vias (10^5)	cpu (min)	#vn	vias (10^5)	cpu (min)	#vn	vias (10^5)	cpu (min)	#vn	vias (10^5)	cpu (m)	#vn	vias (10^5)	cpu (min)
adaptec1	4	16.72	14.82	4	16.72	13.49	4	16.72	12.24	4	16.72	8.93	4	16.72	7.74
adaptec2	0	18.33	12.86	0	18.33	10.93	0	18.33	9.50	0	18.33	5.93	0	18.33	4.94
adaptec3	5	33.00	60.03	5	33.00	52.70	5	33.00	44.31	5	33.00	27.11	5	33.00	22.03
adaptec4	4	30.90	41.11	4	30.90	33.57	4	30.90	29.09	4	30.90	19.16	4	30.90	15.97
adaptec5	0	49.43	53.53	0	49.43	46.61	0	49.43	40.44	0	49.43	27.41	0	49.43	22.83
newblue1	6	21.43	4.93	6	21.43	4.28	6	21.43	3.96	6	21.43	2.87	6	21.43	2.49
newblue2	1	28.18	11.79	1	28.18	9.54	1	28.18	8.82	1	28.18	6.58	1	28.18	5.72
newblue3	1	29.08	29.48	1	29.08	26.94	1	29.08	24.39	0	29.08	15.54	1	29.08	12.33
newblue4	0	44.77	20.23	0	44.77	17.51	0	44.77	16.05	0	44.77	12.10	0	44.77	10.32
newblue5	0	80.30	77.93	0	80.30	65.93	0	80.30	59.94	0	80.31	42.17	0	80.30	35.94
newblue6	0	71.12	33.27	0	71.12	28.66	0	71.12	27.33	0	71.12	19.86	0	71.12	16.90
newblue7	0	157.50	354.41	0	157.50	230.70	0	157.51	204.72	0	157.51	136.66	0	157.51	114.43
bigblue1	0	17.65	19.71	0	17.65	17.86	0	17.65	15.65	0	17.65	11.79	0	17.65	9.71
bigblue2	0	40.34	16.25	0	40.34	13.59	0	40.34	11.72	1	40.34	9.31	1	40.34	8.09
bigblue3	0	50.66	233.01	0	50.66	157.37	0	50.66	112.66	0	50.66	62.26	0	50.66	49.48
bigblue4	0	104.93	301.91	0	104.93	187.24	0	104.93	157.38	0	104.93	102.57	0	104.93	86.32
Sum	21			21			21			21			22		
Ratio		1	1		1.000000	0.824		1.000018	0.727		1.000020	0.504		1.000038	0.424

global-antenna-violations yet was unaware of local-antenna-violation. NVM and AALA were performed on our machine. The routing results of COLA and LAVA are quoted from [2]. As the performing machine of COLA and LAVA is different than AALA and NVM, the runtimes of COLA and LAVA are normalized by the clock rate. The number of overflows is not listed since the results of all layer assignment algorithms have the same number of overflows. Table IV reveals that the results of LAVA contain a lot of nets with antenna violations while AALA can effectively reduce the number of nets with antenna violations (#vn). In 16 benchmarks, AALA can yield ten violation-free assignments while the other works yield no violation-free assignment. As for the total number of antenna violations in all benchmarks, this paper, COLA, NVM, and LAVA yield 21, 43506, 41261, and 29671 antenna violations, respectively. In addition, the via count of AALA is less than COLA and LAVA by 4.9% and 4.6%, respectively. However, AALA consumes more runtime than other algorithms.

Compared to previous works, the proposed method is relatively slow but powerful. We think the proposed method can be treated as the last weapon to tackle the antenna violations after running the entire design flow. For example, suppose a fast antenna-aware layer assignment algorithm is adopted as a default stage in the design flow. If a design gets some unsolved antenna violations after detailed routing, a necessary resynthesis flow starting at P&R or even at logic synthesis is very time-consuming. The proposed method can then be used to remove the antenna violations at this time such that the resynthesis can converge very quickly. As compared to the long time of the resynthesis that may be over several days or weeks, the required runtime of the proposed method is relatively small.

Since previous works do not consider eliminating local antenna violation, for fair comparison, we implement LAVA ($LAVA_O$) and an extended version of LAVA ($LAVA_E$) to follow the strict antenna rule. Since the source code of LAVA is unavailable and $LAVA_O$ is built from scratch, the results of LAVA [2] and $LAVA_O$ are different. Generally $LAVA_O$ generates little more antenna violations than LAVA [2]. $LAVA_O$ assigns each singe net by a two-step algorithm. The two-step algorithm first splits a net into several sub-nets under the constraint $L/n \leq A_{max}$, where L denotes the length of the sub-net and n denotes the number of pins connected by the sub-net, and then assigns each sub-net to the corresponding layer under wire congestion constraints. In $LAVA_E$, the two-step algorithm is modified to split a net into sub-nets under the constraint of the strict antenna rule ($L \leq A_{max}$). Table V reveals that $LAVA_E$ can get the results with fewer antenna violations than $LAVA_O$ at the cost of increased vias and runtime. In addition, although $LAVA_E$ in Table V and AALA in Table IV both obey the strict antenna rule to eliminate local-antenna-violations, $LAVA_E$ still yields more antenna violations than AALA because of the different essences in two algorithms.

B. Effects of Proposed Acceleration Techniques

In Table VI, $AALA_1$, $AALA_2$, and $AALA_3$ represent AALA with the intuitive pruning method, AALA with DPIAP and AALA with heuristic-DPIAP, respectively. The *sepCost* and *viaCost* in $AALA_1$, $AALA_2$ and $AALA_3$ are set to 1 and 100, respectively. As DPIAP accelerates the solution pruning process in AS-SLA and do not prune the non-inferior assignments, the runtime of $AALA_1$ is less than $AALA_2$ by 17.6% and they get the same assignment quality in #vn and vias. The last three major columns in Table VI respectively show the results of $AALA_3$ with different values of τ [in (10)]. When τ is set to 2, 6, and 10, $AALA_3$ can respectively improve the runtimes by 27.3%, 49.6% and 57.6% than $AALA_1$ and yields similar #vn and vias to $AALA_1$. Note that, if τ is larger than ten, the #vn and vias will increase while obtaining slight runtime improvement. Considering runtime reduction issue, divide-and-conquer can be applied to further diminish the runtime of acceleration techniques.

TABLE VII
SEPARATOR NUMBER OF AALA WITH THE DIFFERENT VALUES OF *sepCost*

Benchmark	$AALA_3$ (*sepCost*=0)				$AALA_3$ (*sepCost*=1)				$AALA_3$ (*sepCost*=100)				$AALA_3$ (*sepCost*=500)			
	#vn	#sep (10^5)	vias (10^5)	cpu (min)	#vn	#sep (10^5)	vias (10^5)	cpu (min)	#vn	#sep (10^5)	vias (10^5)	cpu (min)	#vn	#sep (10^5)	vias (10^5)	cpu (min)
adaptec1	4	9.52	16.72	7.12	4	7.55	16.72	8.93	2	1.51	17.90	11.20	2	0.94	18.99	13.39
adaptec2	0	6.18	18.33	4.25	0	4.70	18.33	5.93	0	0.27	18.90	6.35	1	0.17	19.07	6.95
adaptec3	5	33.95	32.97	18.81	5	25.76	33.00	27.11	6	4.99	35.71	33.23	8	3.72	38.31	38.83
adaptec4	4	26.91	30.88	13.58	4	20.22	30.90	19.16	4	1.49	32.87	25.19	5	1.09	33.61	28.57
adaptec5	0	32.40	49.41	20.81	0	25.96	49.43	27.41	0	4.45	52.75	30.87	0	2.89	55.96	34.82
newblue1	6	2.68	21.43	2.28	6	1.93	21.43	2.87	6	0.21	21.82	3.44	5	0.07	22.03	3.67
newblue2	1	9.58	28.18	5.35	1	7.43	28.18	6.58	0	0.68	29.59	8.01	0	0.27	30.32	8.76
newblue3	0	18.63	29.06	10.59	1	15.35	29.08	15.54	0	2.02	31.27	17.11	0	1.28	32.78	18.24
newblue4	0	15.85	44.76	9.27	0	12.10	44.77	12.10	0	0.95	46.84	14.35	0	0.53	47.59	15.72
newblue5	0	36.79	80.28	32.95	0	28.16	80.31	42.17	0	2.92	84.32	47.00	0	1.34	87.27	54.80
newblue6	0	23.85	71.11	15.76	0	19.44	71.12	19.86	0	2.84	74.68	22.47	0	1.08	77.89	24.05
newblue7	0	47.17	157.47	107.31	0	36.48	157.51	136.66	0	2.86	162.83	148.52	0	1.85	164.59	162.87
bigblue1	0	12.07	17.65	9.77	0	9.81	17.65	11.79	0	2.87	19.20	12.74	0	1.54	22.26	14.27
bigblue2	0	7.78	40.33	8.49	0	6.00	40.34	9.31	0	1.28	41.73	12.66	0	0.62	42.94	14.97
bigblue3	0	24.87	50.65	34.27	0	19.30	50.66	62.26	0	0.73	52.40	53.79	0	0.50	52.81	59.52
bigblue4	0	37.69	104.91	72.65	0	29.11	104.93	102.57	0	2.05	108.88	105.13	0	1.28	110.26	116.18
Sum	20				21				18				21			
Ratio		1	1	1		0.777	1	1.339		0.1	1.052	1.524		0.057	1.094	1.705

C. Separators Minimization

To demonstrate that AALA can easily control the separators number by adjusting *sepCost*, Table VII shows the separators number (#sep) of $AALA_3$ with the different values of *sepCost* when *viaCost* is set to 100 and τ is set to 6. This experiment reveals that the separators can be significantly reduced if the value of *sepCost* increases. $AALA_3$ with sepCost = 500 can reduce 94.3% separators than that with sepCost = 0 and only increases the via count by 9.4%. However, the runtime of AALA increases as the value of *sepCost* increases.

VII. CONCLUSION

This paper presents an antenna-safe single-net layer assignment called AS-SLA that can identify the minimum-cost antenna-safe layer assignment solution for a net, and can simultaneously minimize the via count and separators. Moreover, a dynamic-programming-based inferior assignment pruning algorithm and a heuristic solutions pruning scheme are proposed to accelerate AS-SLA. Finally, this paper presents an AALA adopting AS-SLA, AALA not only avoids global-antenna-violations but also eliminates local-antenna-violations. As compared to previous works, AALA can reduce the number of antenna violations significantly.

REFERENCES

[1] D. Wu, J. Hu, and R. Mahapatra, "Antenna avoidance in layer assignment," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 25, no. 4, pp. 1643–1656, Apr. 2006.

[2] T.-H. Lee and T.-C. Wang, "Simultaneous antenna avoidance and via optimization in layer assignment of multilayer global routing," in *Proc. Int. Conf. Computer-Aided Design*, 2010, pp. 312–318.

[3] H. Shin, C.-C. King, and C. Hu, "Thin oxide damage by plasma etching and ashing process," in *Proc. Int. Reliab. Phys. Symp.*, 1992, pp. 37–41.

[4] H. Watanabe, J. Komori, K. Higashitani, M. Sekine, and H. Koyama, "A wafer level monitoring method for plasma-charging damage using antenna PMOSFET Test Structure," *IEEE Trans. Semicond. Manuf.*, vol. 10, no. 2, pp. 228–232, May 1997.

[5] X. Gao and L. Macchiarulo, "A jumper insertion algorithm under antenna ratio and timing constraints," in *Proc. Int. Conf. Computer-Aided Design*, 2011, pp. 290–297.

[6] M. Moffitt, "MAIZEROUTER: Engineering an effective global router," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 27, no. 11, pp. 2017–2026, Nov. 2008.

[7] M. Cho, K. Lu, K. Yuan, and D. Z. Pan, "BoxRouter 2.0: A hybrid and robust global router with layer assignment for routability," *ACM Trans. Design Autom. Electr. Syst,* vol. 14, no. 2, article 32, 2009.

[8] M. M. Ozdal and M. D.F. Wong, "ARCHER: A history-driven global routing algorithm," in *Proc. Int. Conf. Computer-Aided Design*, 2007, pp. 488–495.

[9] Y.-J. Chang, Y.-T. Lee, J.-R. Gao, P.-C. Wu, and T.-C. Wang, "NTHU-Route 2.0: A robust global router for modern design," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 29, no. 12, pp. 1931–1944, Dec. 2010.

[10] M. Pan and C. Chu, "FastRoute: A step to integrate global routing into placement," in *Proc. Int. Conf. Computer-Aided Design*, 2006, pp. 464–471.

[11] Y. Xu, Y. Zhang, and C. Chu, "FastRoute 4.0: Global router with efficient via minimization," in *Proc. Asia South Pacific Design Autom. Conf*., 2009, pp. 576–581.

[12] J. Hu, J. Roy, and I. Markov, "Completing high-quality global routes," in *Proc. Int. Symp. Phys. Design*, 2010, pp. 35–41.

[13] K.-R. Dai, W.-H. Liu, and Y.-L. Li, "NCTU-GR: Efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-D global routing," *IEEE Trans. Very Large Scale (VLSI) Syst.*, vol. 20, no. 3, pp. 459–472, Mar. 2012.

[14] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "NCTU-GR 2.0: Multi-threaded collision-aware global routing with bounded-length maze routing," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 32, no. 5, pp. 709–722, May 2013.

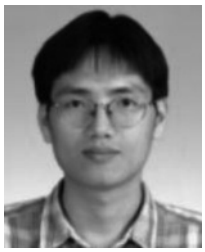
[15] W.-H. Liu, Y.-L. Li, and C.-K. Koh, "A fast maze-free routing congestion estimator with hybrid unilateral monotonic routing," in *Proc. Int. Conf. Computer-Aided Design*, 2012, pp. 713–719.

[16] N. J. Naclerio, S. Masuda, and K. Nakajima, "The via minimization problem is NP-complete," *IEEE Trans. Comput.*, vol. 38, no. 11, pp. 1604–1608, Nov. 1989.
[17] T.-H. Lee and T.-C. Wang, "Congestion-constrained layer assignment for via minimization in global routing," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 27, no. 9, pp. 1643–1656, Sep. 2008.
[18] T.-H. Lee and T.-C. Wang, "Robust layer assignment for via optimization in multilayer global routing," in *Proc. Int. Symp. Phys. Design*, pp. 159–166, 2009.
[19] W.-H. Liu and Y.-L. Li, "Negotiation-based layer assignment for via count and via overflow minimization," in *Proc. Asia South Pacific Design Autom. Conf.*, 2011, pp. 539–544.
[20] J. Ao, S. Dong, S. Chen, and S. Goto, "Delay-driven layer assignment in global routing under multitier interconnect structure," in *Proc. Int. Symp. Phys. Design*, 2013, pp. 101–107.
[21] Y. Wei, Z. Li, C. Sze, S. Hu, C. Alpert, and S. Sapatneker, "CATALYST: Planning layer directives for effective design closure," in *Proc. DATE*, 2013, pp. 1873–1878.
[22] Sigda. (2008). "ISPD08 global routing contest website" [Online]. Available: http://archive.sigda.org/ispd2008/contests/ispd08rc.html
[23] W.-H. Liu and Y.-L. Li, "Optimizing the antenna area and separators in layer assignment of multilayer global routing," in *Proc. Int. Symp. Phys. Design*, 2012, pp. 137–144.

Wen-Hao Liu received the B.S. and Ph.D. degrees in computer science from National Chiao Tung University, Hsinchu, Taiwan.

He is currently a Post-Doctoral Fellow at National Tsing Hua University, Hsinchu. He has developed a well-known global router called NCTU-GR that was selected to be the evaluation tool for DAC12 and ICCAD12 placement contests. In the last five years, he has published three journal papers for IEEE Transactions, three DAC papers, two ICCAD papers, one DATE paper, three ISPD papers, and four ASP-DAC papers. His current research interests include global routing, placement, and clock network synthesis.

Dr. Liu was a winner of the ISPD Clock Network Synthesis Contest in 2009. He is currently serving on the review boards of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, *ACM Transactions on Design Automation of Electronic Systems*, and *Integration, the VLSI Journal*. He is also the co-organizer for the ICCAD13 Cadathlon contest and the ISPD14 placement contest, and serves on the technical program committees and the best paper selection committees of ASP-DAC 2014.



Yih-Lang Li (M'11) received the B.S. degree in nuclear engineering, and the M.S. and the Ph.D. degrees in computer science, majoring in designing and implementing a highly parallel cellular automata machine for fault simulation, all from the National Tsing Hua University, Hsinchu, Taiwan.

In 2003, he joined the Faculty of the Department of Computer Science, National Chiao Tung University (NCTU), Hsinchu, where he is currently an Associate Professor. Prior to joining NCTU, from 1995 to 1996 and from 1998 to 2003, he was a Software Engineer and an Associate Manager with Springsoft Corporation, Hsinchu, where he first completed the development of DRC tool for the custom-based layout design, and then established and lead a routing team for developing a block-level shape-based router for custom-based layout design. His current research interests include physical synthesis, parallel architecture, and VLSI testing.

Dr. Li joined the technical committee of the first CAD contest in Taiwan and had served as a Committee member for ten years. He also served as the Contest Chair of the first CAD contest at ICCAD in 2012.