An Optimal Jumper-Insertion Algorithm for Antenna Avoidance/Fixing

Bor-Yiing Su and Yao-Wen Chang, *Member, IEEE*

***Abstract*—As the process technology enters the nanometer era, reliability has become a major concern in the design and manufacturing of very large-scale integration circuits. In this paper, we focus on one reliability issue—jumper insertion in routing trees for avoiding/fixing antenna-effect violations at the routing/postlayout stages. We formulate the jumper insertion for antenna avoidance/fixing as a tree-cutting problem and present the first optimal algorithm for the tree-cutting problem. We show that the tree-cutting problem exhibits the properties of optimal substructures and greedy choices. With these properties, we present an $O(V)$-time optimal jumper-insertion algorithm that uses the minimum number of jumpers to avoid/fix the antenna violations in a routing tree with V vertices. Experimental results show the superior effectiveness and efficiency of our algorithm.**

***Index Terms*—Physical_design, reliability, routing.**

I. Introduction

As THE PROCESS technology enters the nanometer era, product reliability and manufacturing yield have become major concerns in the design and manufacturing of very large-scale integration circuits. The fine feature size of modern IC technologies is typically achieved by using plasma-based processes. In nanometer technology, more stringent process requirements cause some advanced high-density plasma reactors adopted in the production lines to achieve fine-line patterns [5]. However, these plasma-based processes will charge conducting components of a fabricated structure. As a result, the accumulated charges may affect the quality of ICs. This is called the antenna effect.

During metallization, long floating interconnects act as temporary capacitors and accumulate charges gained from the energy provided by fabrication steps such as plasma etching. A random discharge of the floating node due to subsequent process steps could permanently damage transistors in the IC [8], [9]. For instance, the exposed polysilicon and metal structures connected to a thin-oxide transistor will collect charge from the processing environment (e.g., reactive-ion etch) and damage the transistor when the discharging current flows through the thin oxide. The mechanism of antenna damage is not fully understood, but there is experimental evidence indicating when charging occurs and how it may affect the quality of gate oxide [8], [9]. Charging occurs when conductor layers, which are not covered by a shielding layer of oxide, are directly exposed to plasma. The amount of such charging is proportional to this plasma-exposed area. If conductor layers are connected to a diffusion layer pattern, such charges are discharged to the substrate through the diffusion; see Fig. 1(b)–(d) for illustrations. On the other hand, if the charged conductor layers are connected only to the gate oxide, Fowler–Nordheim (F–N) tunneling current through thin-oxide discharges such charges and causes damage to the thin oxide [8]; see Fig. 1(b) and (c). As shown in Fig. 1, interconnects are manufactured layer by layer. Before a conducting path to the diffusion is formed in metal-2 layer-pattern etching [see Fig. 1(d)], the interconnects in the poly- and metal-1 layers might have accumulated so many charges that they cause damage on the gate, as shown in the left-hand side of Fig. 1(c) (note that there will not be any antenna violation after a conducting path to the diffusion is formed).

There are three kinds of solutions to reduce the antenna effect [2] as follows.

1) Jumper insertion: Break the signal wires with antenna violations and route them to the highest layers by jumper insertion. This reduces the charge amount for violated wires during manufacturing.
2) Embedded protection diode: Add protection diodes on every input port of a standard cell.
3) Diode insertion during layout design: Fix those wires with antenna violations that have enough rooms for "under-the-wire" diode insertion. During wafer manufacturing, all the inserted diodes are floating (or ground). A diode can be used to protect all input ports that are connected to the same output ports.

Comparing the three methods, for the second method of embedded protection diode, since these diodes are embedded and fixed, they might consume unnecessary areas when there is no violation at the connecting wire. For the third method, we need extra silicon area to place the diodes. Because the number of diodes needed for fixing antenna violations grows dramatically as the feature size shrinks, it is often hard to preserve enough space for diodes in nanometer IC designs. As a result, jumper insertion becomes a very popular approach for avoiding/fixing antenna violations. The function of jumper insertion can be explained using Fig. 2. In Fig. 2(a), when the metal-1 layer is manufactured, the gate on the right might be damaged because the large area of the metal-1 interconnection can accumulate sufficient charges to damage the gate. However,

Manuscript received September 22, 2005; revised April 2, 2006 and September 13, 2006. The work of B.-Y. Su and Y.-W. Chang was supported in part by the National Science Council of Taiwan, R.O.C., under Grants NSC 93-2815-C-002-046-E, NSC 94-2215-E-002-005, and NSC 94-2752-E-002-008-PAE. This paper was recommended by Associate Editor T. Yoshimura.

B.-Y. Su is with the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: b90901130@ntu.edu.tw).

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

Digital Object Identifier 10.1109/TCAD.2007.896307

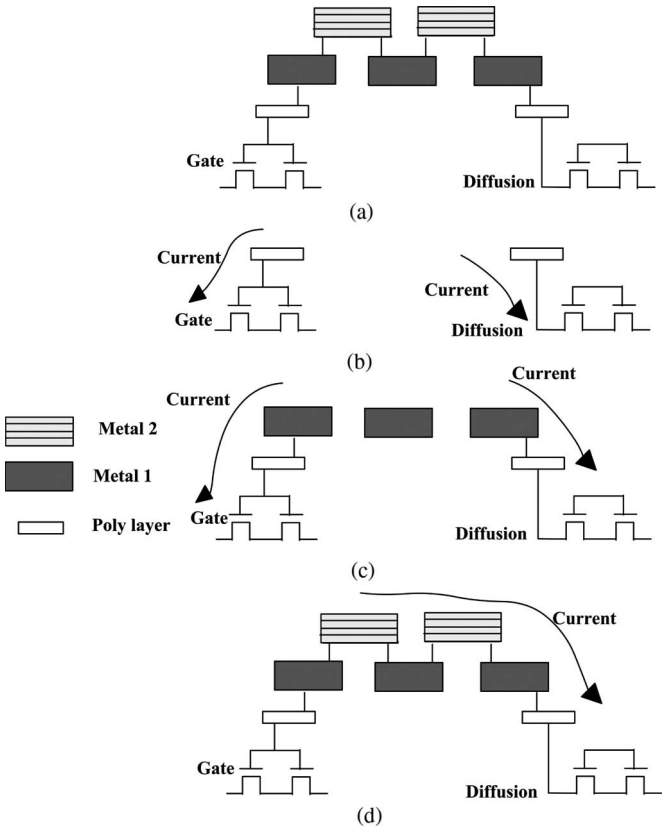

Fig. 1. Antenna effect. (a) Example routing. (b) Late stage of polylayer-pattern etching of (a). Charge on the left polypattern is discharged through the gate, while charge on the right is discharged through the diffusion. (c) Late stage of metal-1 layer-pattern etching of (a). Charge on the left metal-1 pattern is discharged through the gate, while charge on the right is discharged through the diffusion. (d) Late stage of metal-2 layer-pattern etching. Charges on all the metal-2 patterns are discharged through the diffusion.

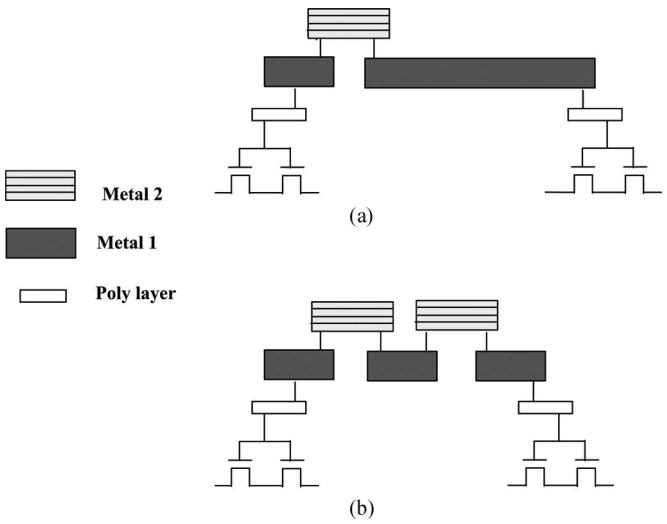

Fig. 2. Jumper insertion. (a) Stage before inserting a jumper. (b) Stage after inserting a jumper from the metal-1 layer to the metal-2 layer.

if we insert a jumper to route the interconnect on the metal-2 layer, as shown in Fig. 2(b), the effective conductor layer becomes smaller. Therefore, the stored charge is not enough to damage the gate on the right, and thus, we can avoid the antenna violation.

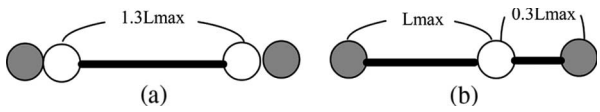

Fig. 3. Jumper insertion for a wire of 1.3 $L_{\max}$ long. (a) Two jumpers are needed for fixing the antenna violation if jumpers can be inserted only beside gate terminals. (b) One jumper suffices to fix the problem if it can be inserted at an arbitrary position of the wire.

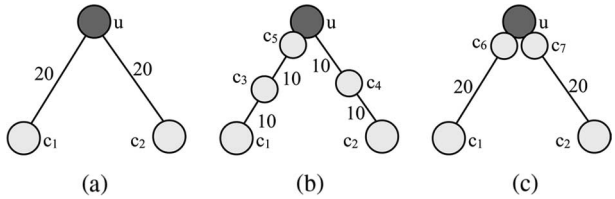

Fig. 4. (a) Routing tree with one sink node u_1. (b) Work in [10] needs jumpers c_3, c_4, and c_5 to solve the antenna violations. (c) This paper needs only two jumpers, c_6 and c_7, to satisfy the antenna rule.

Although jumper insertion is currently the most popular approach for antenna avoidance/fixing, jumpers induce vias that will consume silicon areas and reduce circuit performance. Therefore, it is desired to fix antenna violations by using the least jumpers. Recently, Ho *et al.* [4] proposed a bottom-up approach to insert jumpers in a routing tree for antenna avoidance. The work inserts jumpers only beside gate terminals, and its optimality holds only for this special condition of inserting jumpers right beside gate terminals. As an example, as shown in Fig. 3, the wire segment is of 1.3 $L_{\max}$ long, where $L_{\max}$ denotes the upper bound for antenna (i.e., any wire longer than $L_{\max}$ will violate the antenna rule). For this wire segment, the work in [4] needs two jumpers to fix the antenna violation [see Fig. 3(a)], while a single jumper suffices to fix the violation [see Fig. 3(b)].

Another recent work by Wu *et al.* [10] extends the work in [4] to handle the problem. With the implementation scheme proposed by Kundu and Misra [6], the work in [10] can achieve the linear-time complexity for jumper insertion in a routing tree for antenna avoidance/fixing. To fix the antenna violation of a sink node (a gate terminal in this paper), the work first removes all subtrees around the node that violate the antenna rules. After all such subtrees are removed, if the sink still violates the antenna rule, the work will continually remove the heaviest branch from the sink until the antenna rules are satisfied. The approach in the study in [10] is not optimal under some special cases. As the routing-tree example shown in Fig. 4(a), u is a sink node that violates the antenna rules, and c_1 and c_2 are two inserted jumpers. The number beside each edge denotes the antenna charge weight (measured by the antenna-strength-to-gate-size ratio, the wire length, the wire area, and/or the wire perimeter), and the maximum antenna weight that a sink node can bear is assumed to be ten. For the work in [10], the algorithm will remove all of the subtrees around u. Thus, it inserts jumpers c_3 and c_4. However, u still violates the antenna rules. As a result, the algorithm will insert jumper c_5 at the heaviest branch $e(u, c_3)$, as shown in Fig. 4(b). For this case, nevertheless, two jumpers suffice to solve the antenna violations; see the jumpers c_6 and c_7 shown in Fig. 4(c).

In this paper, we consider the general case of inserting jumpers at arbitrary positions (e.g., in any position of a tree edge). We formulate the general jumper insertion for antenna avoidance (applicable at the routing stage) and/or fixing (applicable at the postlayout stage) as a tree-cutting problem. We show that the tree-cutting problem exhibits the properties of optimal substructures and greedy choices. With these properties, a greedy algorithm suffices to find an optimal solution [3]. Based on the theory, we present an $O(V)$-time optimal jumper-insertion algorithm that uses the minimum number of jumpers to fix the antenna violations in a routing tree with V vertices. Compared with the previous work in [4] and four types of heuristics, our algorithm outperforms those methods by large margins. Experimental results based on the layouts generated from the multilevel routers in the study in [1] and [7] on a set of commonly used Microelectronics Center of North Carolina (MCNC) benchmarks show that our algorithm can significantly reduce the number of antenna violations.

The remainder of this paper is organized as follows. Section II formulates the problem of jumper insertion on a routing tree for avoiding/fixing antenna violations. Section III presents an optimal algorithm for the proposed problem. Section IV proves the optimality of the algorithm. Section V analyzes the complexity of the algorithm. Section VI extends the algorithm to the handling of Steiner trees and obstacles. Section VII reports the experimental results. Finally, the conclusions and future work are given in Section VIII.

II. Problem Definition

To avoid/fix the antenna violation, we require that the total effective conductor connecting to a gate be less than or equal to a threshold $L_{\max}$. The threshold could be the wire-length limit, the wire-area limit, the wire-perimeter limit, the ratio of antenna strength (length, area, perimeter, etc.) to the gate size, or any model of the strength of antenna effect caused by conductors. Typically, a net is modeled as a routing tree, where a node in the tree denotes a circuit terminal (a gate or a diffusion) and an edge denotes the interconnection between two circuit terminals. Since the interconnection connecting to a diffusion terminal will not cause any antenna violation, as explained in Section I, we shall focus on those connecting to gate terminals.

Let $T = (V, E)$ be a routing tree, the set V of nodes represents all gate terminals, the set E of edges denotes the wires connecting the circuit terminals, and an edge weight gives the measure of the wires with the same unit as $L_{\max}$. A gate will violate the antenna rule if the effective conductor incident on the gate (i.e., the effective weight—the sum of the weights of the edges incident to the corresponding node) is larger than $L_{\max}$. To reduce the antenna effects on a gate, we can apply the technique illustrated in Fig. 2 by adding a jumper on a wire connecting to the gate to reduce the effective conductor. This operation is modeled as adding a cutting node on the tree edge corresponding to the wire to reduce the effective edge weight associated with the gate node. As aforementioned, jumpers are implemented by vias which will consume silicon areas and reduce circuit performance. Therefore, it is desired to fix antenna violations by using the least jumpers. In other words, given a routing tree $T = (V, E)$ and an upper bound on the antenna $L_{\max}$, we intend to add the minimum number of cutting nodes so that the effective edge weight associated with each node is smaller than $L_{\max}$. Let $L(u)$ denote the sum of edge weights (wire lengths, wire areas, wire perimeter limit, the ratio of antenna strength, etc.) between the node u and all its neighbors. We formulate the problem of jumper insertion on a routing tree for antenna avoidance/fixing as a tree-cutting problem as follows:

- Jumper Insertion on a Routing Tree for Antenna Avoidance (Problem JITA): Given a routing tree $T = (V, E)$ and an upper bound $L_{\max}$, find the minimum set C of cutting nodes, $c \neq u$ for any $c \in C$ and $u \in V$, so that $L(u) \leq L_{\max}, \forall u \in V$.

```
Algorithm: BUJI(T, L_max, t_total, C)
Input: T = (V, E) /* The given tree.  */
       L_max /* Upper Bound on antenna */
       C /* Cutting set */
       t_total /* Total Edge Length in T */
1   while (t_total > L_max)
2       for each leaf node u ∈ T not having been processed
3           Mark u as processed;
4           if l(u, p(u)) > L_max
5               if u ∉ C
6                   Let c be the node between u and p(u) with
                    l(c, u) = L_max and l(c, p(u)) = l(u, p(u)) − L_max;
7                   C ← C ∪ {c};
8                   t_total ← t_total − L_max;
9                   T(V, E) ← T(V ∪ {c} \ {u}, E \ {e(u, c)});
10      for each subleaf node u_p ∈ T
            Let u_1, u_2, ..., u_k denote all children nodes of u_p.
11          totallen ← Σ_{i=1}^{k} l(u_p, u_i);
12          if totallen ≤ L_max
13              LessEqual(T, t_total, C, u_p, totallen, L_max) ;
14          else
15              More(T, t_total, C, u_p, totallen, L_max);
```

Fig. 5. Algorithm BUJI deals with the leaf nodes first, and then call Subroutines LessEqual and More to deal with the subleaf nodes.

III. Algorithm for Finding the Minimum $|C|$

For the JITA problem, we present in this section an $O(V)$-time optimal algorithm, named Bottom-Up Jumper Insertion (BUJI), for finding the minimum cutting set C for a given routing tree $T = (V, E)$ with V nodes (note that we use V to denote the set or the number of nodes in a routing tree). Algorithm BUJI is summarized in Fig. 5. To simplify the presentation, we assume that the antenna bound $L_{\max}$ is measured by wire length. Let $l(e)$ [or $l(u, v)$] be the length (i.e., weight) of the edge $e = (u, v)$ in T. Let $p(u)$ denote node u's parent. In the BUJI algorithm, we add the cutting nodes into the original tree in a bottom-up manner. We first define a subleaf node as follows.

Definition 1: A subleaf is a node for which all its children are leaf nodes, and all the edges between it and its children have lengths $\leq L_{\max}$.

We derive the algorithm based on the following two steps.

Step 1) (lines 2–8 of Algorithm BUJI): Deal with every leaf node. In this case, our main goal is to prevent every

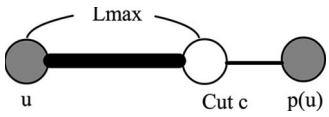

Fig. 6. Explanation of lines 5–8 in the BUJI algorithm.

Subroutine: LessEqual(T, t_{total}, C, u_p, $totallen$, L_{max})
1 **if** $p(u_p)$ does not exist
2 **return**
3 **if** $totallen + l(u_p, p(u_p)) \leq L_{max}$
4 $t_{total} \leftarrow t_{total} - totallen$;
5 $T(V, E) \leftarrow T(V \setminus \cup_{i=1}^{k}\{u_i\}, E \setminus \cup_{i=1}^{k}\{e(u_i, u_p)\})$;
6 **else**
Let c be the node on $e(u_p, p(u_p))$
with $l(c, u_p) + totallen = L_{max}$;
7 $C \leftarrow C \cup \{c\}$;
8 $t_{total} \leftarrow t_{total} - L_{max}$;
9 $T(V, E) \leftarrow T(V \setminus (\cup_{i=1}^{k}\{u_i\}) \cup \{c\} \setminus \{u_p\},$
$E \setminus (\cup_{i=1}^{k}\{e(u_i, u_p)\}) \setminus \{e(u_p, c)\})$;

Fig. 7. Case when $\text{totallen} \leq L_{\max}$.

leaf node from antenna violation. Obviously, if we have dealt with a leaf node, we need not consider it any more. Therefore, line 3 of the BUJI algorithm marks these nodes to ensure that every leaf node is processed only once. If $l(u, p(u)) \leq L_{\max}$, the leaf node u satisfies the antenna rule, and thus, we do nothing. However, if $l(u, p(u)) > L_{\max}$, we might have to add cutting nodes to satisfy that $L(u) \leq L_{\max}$. For this case, we can further divide it into two subcases as follows.

1) $u \in C$: In this subcase, since u is not a gate terminal, we need not insert any cutting node to prevent it from violating antenna rules.
2) $u \notin C$: In this subcase, we need at least one cutting node to prevent u from antenna violation. We claim that $l(u, c) = L_{\max}$ (and thus $l(c, p(u)) = l(u, p(u)) - L_{\max}$) gives the best position for inserting the cutting node (the proof is given in the next section; see Fig. 6 for an illustration). Therefore, we add c into C, add c into V, and cut the node u and edge $e(u, c)$ from the original tree T (lines 5–8).

Step 2) (lines 9–14 of BUJI): Deal with every subleaf node. In this case, our main goal is to prevent every subleaf node from antenna violation. Moreover, we delete some nodes and edges to make each subleaf node as a leaf node. We classify the subleaf nodes into two categories by the sum of lengths between the node and its children. Let u_p be a subleaf node and u_i, $\forall 1 \leq i \leq k$ be its children. Let $\text{totallen} = \sum_{i=1}^{k} l(u_i, u_p)$.

Case 2.1) $\text{totallen} \leq L_{\max}$: For this case, we apply the LessEqual subroutine; see Fig. 7. If u_p and its children form an isolated component, they must satisfy the antenna rule, and thus, we are done with the subroutine. If $\text{totallen} + l(u_p, p(u_p)) \leq L_{\max}$, u_p will not violate the antenna rule. Therefore, we simply cut u_p's children from the original tree to make u_p as a leaf node (lines 3–5 in LessEqual). Otherwise, we must add at least one cutting node c to prevent u_p from antenna violation. We claim that $l(c, u_p) + \text{totallen} = L_{\max}$ gives the best position for inserting the cutting node; see Fig. 8(a). Therefore, we add c into C, and cut u_p and all its children from the original tree T to make c as a leaf node (lines 6–9 in LessEqual).

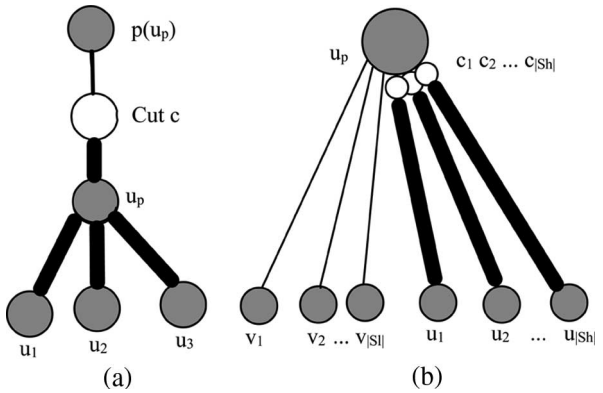

Fig. 8. (a) Illustration of the LessEqual subroutine. Here, $l(u_p, u_1) + l(u_p, u_2) + l(u_p, u_3) + l(u_p, c) = L_{\max}$. (b) Illustration of the More subroutine.

Subroutine: More(T, t_{total}, C, u_p, $totallen$, L_{max})
1 $S = \cup_{i=1}^{k}\{l(e(u_i, u_p))\}$;
2 $S_h = SPLIT(S, L_{max})$;
Let c_i be the nodes between u_p and u_i with $l(c_i, u_p) = 0$,
$l(c_i, u_i) = l(u_p, u_i)$ and $l(e(u_i, u_p)) \in S_h$ $\forall 1 \leq i \leq |S_h|$;
3 $C \leftarrow C \cup \{c_i\}$ $\forall\ 1 \leq i \leq |S_h|$;
4 $T(V, E) \leftarrow T(V \setminus \cup_{i=1}^{|S_h|}\{u_i\}, E \setminus \cup_{i=1}^{|S_h|}\{e(u_i, u_p)\})$;
5 $minuslen \leftarrow \sum_{s \in S_h} s$;
6 $t_{total} \leftarrow t_{total} - minuslen$;
7 $LessEqual(T, t_{total}, C, u_p, totallen - minuslen, L_{max})$;

Fig. 9. Case when $\text{totallen} > L_{\max}$.

Case 2.2) $\text{totallen} > L_{\max}$: For this case, we apply the More subroutine; see Fig. 9. We first introduce the set $S = \cup_{i=1}^{k}\{l(e(u_i, u_p))\}$ from the subleaf node u_p and its k children. Then, we apply the linear-time algorithm SPLIT, as presented in the study in [6], to split the set S into two disjoint subsets, S_h and S_l, where S_h is the higher subset and S_l is the lower subset (to make this paper self-contained, we also give the SPLIT algorithm in Fig. 10). The two subsets have three important properties: 1) for any $a \in S_l$ and any $b \in S_h$, we have $a \leq b$; 2) $\sum_{s \in S_l} s \leq L_{\max}$; and 3) for any $b \in S_h$, we have $\sum_{s \in S_l} s + b > L_{\max}$. Moreover, the SPLIT algorithm will return the S_h subset. We claim that c_i on edge

```
Subroutine: SPLIT(S, Bound) [6]
1  if |S| = 1
2      if Σ_{s∈S} s ≤ Bound
3          return ∅;
4      else
5          return S;
6  else
7      Median-find-and-halve(S) and let S_h be the higher half of S;
8      W = Σ_{s∈S and s∉S_h} s;
9      if W = Bound
10         return S_h;
11     else if W < Bound
12         return SPLIT(S_h, Bound − W);
13     else (W > Bound)
14         return SPLIT(S \ S_h, W) + S_h;
```

Fig. 10. This subroutine returns the required subset S_h from S. Median-find-and-halve (S) finds the median m of set S and partitions S into two subsets S_l and S_h, where each element in S_l is $\leq m$ and each element in S_h is $\geq m$. Moreover, $|S_h| \leq |S_l| \leq |S_h| + 1$.

$e(u_i, u_p)$ with $l(c_i, u_p) = 0$ [and, thus, $l(c_i, u_i) = l(u_p, u_i)$] and $l(e(u_i, u_p)) \in S_h$, $\forall 1 \leq i \leq |S_h|$ gives the best positions for inserting the cutting nodes; see Fig. 8(b). Therefore, we add $c_1, \ldots, c_{|S_h|}$ into C and cut $u_1, \ldots, u_{|S_h|}$ from the original tree T (lines 1–6 in More). Moreover, we call Subroutine LessEqual to further reduce u_p into a leaf node (line 7 in More).

When the total length of the tree T is not larger than $L_{\max}$, Algorithm BUJI terminates, and C is a cutting set of the minimum size.

It should be noted that we focus on the processing of gate terminals (sinks) in the preceding discussions. When the source nodes of the routing tree are encountered, Algorithm BUJI can be applied as usual by treating the source nodes as cutting nodes, since the source nodes play the same role as the cutting nodes for antenna-effect processing.

IV. Proof of the Optimality of $|C|$

Algorithm BUJI is greedy in nature. To prove that Algorithm BUJI finds the optimal cutting set (of the minimum size), we have to show that the JITA problem exhibits optimal substructure and has the greedy-choice property [3]. A problem exhibits optimal substructure if an optimal solution to the problem contains within it optimal solutions to the subproblems; a problem has the greedy-choice property if a globally optimal solution can be arrived at by making a locally optimal (greedy) choice [3].

Theorem 1: The JITA problem exhibits optimal substructure.

Proof: We prove this property by contradiction. Given a tree $T = (V, E)$, suppose that the cutting set C is the optimal solution of the tree. Every cutting node in C cuts the given tree into two subtrees. Let some cutting node $c \in C$ cut T into subtree T_1 and T_2. Let the cutting set $C_1 \subseteq C (C_2 \subseteq C)$ be the set of cutting nodes in $T_1 (T_2)$. Thus, $C = C_1 \cup C_2 \cup \{c\}$. If C_1 does not form an optimal solution on T_1, let C_1' be the optimal solution on T_1, and thus, $|C_1'| < |C_1|$. Let $C' = C_1' \cup C_2 \cup \{c\}$. We have $|C'| = |C_1'| + |C_2| + 1 < |C_1| + |C_2| + 1 = |C|$. This contradicts the assumption of the optimality of set C. Therefore, the JITA problem has optimal substructure. ■

Now, we show that the JITA problem has the greedy-choice property, and Algorithm BUJI finds the best solution in each step. First, we show that Algorithm BUJI has greedy-choice property among all leaf nodes. Then, we show that BUJI has greedy-choice property among all subleaf nodes.

Lemma 1: Lines 2–8 of Algorithm BUJI finds the best cutting set so that every leaf node u satisfies the antenna rule (i.e., $L(u) \leq L_{\max}, \forall$ leaf nodes u).

Proof: If $l(u, p(u)) \leq L_{\max}$, leaf node u must satisfy the antenna rule. Therefore, we do nothing. Otherwise, we must add at least one cutting node to satisfy the antenna rule. Moreover, if $u \in C$, since u is not a gate terminal, we need not insert any cutting node to prevent it from violating antenna rules.

In lines 5–8, we deal with the case when u is not a cutting node. In this case, we must insert at least one cutting node between u and $p(u)$ to prevent u from antenna violation. The possible cutting range is represented by a thick line in Fig. 6. Let the optimal solution add the cutting node c' between u and u_p with $L(u) \leq L_{\max}$ and $L(p(u)) \leq L_{\max}$. Because $l(c, p(u)) = l(u, p(u)) - L_{\max} \leq l(c', p(u))$, if we replace c' with c, the antenna rule that $L(u) \leq L_{\max}$ is satisfied, and the antenna rule that $L(p(u)) \leq L_{\max}$ is more tightly satisfied. Therefore, among all nodes on the cutting range, c is the best position of adding cutting nodes. ■

We proceed to show that lines 9–14 in BUJI finds the best cutting set for each subleaf node u_p. In this step, we classify the subleaf nodes into two categories based on the sum of lengths between u_p and its children u_i: $\sum_{i=1}^{k} l(u_p, u_i) \leq L_{\max}$ and $\sum_{i=1}^{k} l(u_p, u_i) > L_{\max}$. Therefore, we show that each case is with the greedy-choice property, and we find the best cutting set in each case.

Lemma 2: Subroutine LessEqual finds the best cutting set so that every subleaf node u_p satisfies the antenna rule [i.e., $L(u_p) \leq L_{\max}, \forall$ subleaf nodes u_p satisfying $\sum_{i=1}^{k} l(u_p, u_i) \leq L_{\max}$, where $u_p = p(u_i)$].

Proof: If u_p and its children form an isolated component, they must satisfy the antenna rule. Therefore, we do nothing. If $\sum_{i=1}^{k} l(u_i, u_p) + l(u_p, p(u_p)) = L(u_p) \leq L_{\max}$ is satisfied, u_p satisfies the antenna rule, and thus, we need no cutting node. As a result, we can directly make u_p as a leaf node without adding any cutting nodes in line 5 of Subroutine LessEqual. Otherwise, because $\sum_{i=1}^{k} l(u_i, u_p) + l(u_p, p(u_p) > L_{\max}$, we need to insert at least one cutting node to maintain $L(u_p) \leq L_{\max}$. Therefore, the least number of cutting nodes is one. The possible cutting range is represented by a thick line in Fig. 8(a). Suppose that the optimal solution adds cutting node c' other than c to maintain $L(u_p) \leq L_{\max}$ and $L(p(u_p)) \leq L_{\max}$. For the case that c' is on edge $e(u_p, c)$, since $l(c, p(u_p)) = l(u_p, p(u_p)) - l(c, u_p) \leq l(c', p(u_p)) = l(u_p, p(u_p)) - l(c', u_p)$ if we replace c' with c, $L(u_p) \leq L_{\max}$ is satisfied, and $L(p(u_p)) \leq L_{\max}$ is satisfied more tightly. For the case that c' is on some edge between u_p and its children, since $l(c, p(u_p)) = l(u_p, p(u_p)) - l(c, u_p) \leq l(u_p, p(u_p))$ if we replace c' with c, $L(u_p) \leq L_{\max}$ is satisfied,

and $L(p(u_p)) \leq L_{\max}$ is satisfied more tightly. Therefore, c is the best position for adding the cutting node. ■

Lemma 3: Subroutine More finds the best cutting set so that every subleaf node u_p satisfies the antenna rule [i.e., $L(u_p) \leq L_{\max}, \forall$ subleaf nodes u_p satisfying $\sum_{i=1}^{k}(l(u_p, u_i)) > L_{\max}$, where $u_p = p(u_i)$].

Proof: By the proof of the study in [6], we know that the set $S = S_l \cup S_h$ has the following three properties.

1) For any $a \in S_l$ and any $b \in S_h$, we have $a \leq b$.
2) $\sum_{s \in S_l} s \leq L_{\max}$.
3) For any $b \in S_h$, we have $\sum_{s \in S_l} s + b > L_{\max}$.

Using the properties above, we can easily verify that S_l is the set with $\sum_{s \in S_l} s \leq L_{\max}$, and the size of the set is maximized. Moreover, if two or more sets have the same maximum size and satisfy the same summation rule, S_l is the one with the minimum $\sum_{s \in S_l} s$ value. Since $\sum_{s \in S_l} s \leq L_{\max}$, we need no cutting nodes inside the set, but we must add cutting nodes on every edge in S_h. Since S_l's size is maximized, the minimum number of cutting nodes is $|S| - |S_l| = |S_h|$. Moreover, every edge $e(u_i, u_p)$ with $l(e(u_i, u_p)) \in S_h$ needs a cutting node. The cutting range of every edge $e(u_i, u_p)$ is represented by the thick line shown in Fig. 8(b). Let the optimal solution choose the set S'_l (and $S'_h = S \setminus S'_l$) with the optimal size $|S_l|$ such that the optimal solution does not add any jumper on edge $e(u'_i, u_p)$, $\forall l(e(u'_i, u_p)) \in S'_l$. In addition, let the optimal solution select $|S_h|$ cutting nodes $c'_1, c'_2, \ldots, c'_{|S_h|}$. Since $l(c_i, u_p) = 0 \leq l(c'_i, u_p), \forall 1 \leq i \leq |S_h|$, and $\sum_{s \in S_l} s \leq \sum_{s' \in S'_l} s'$, $L(u_p) = \sum_{s \in S_l} s + \sum_{i=1}^{|S_h|} 0 + l(u_p, p(u_p)) \leq L'(u_p) = \sum_{s' \in S'_l} s' + \sum_{i=1}^{|S_h|} l(c'_i, u_p) + l(u_p, p(u_p))$. Thus, if we replace each c'_i by c_0, the antenna rule on u_p is satisfied more tightly. Therefore, $c_1, \ldots, c_{|S_h|}$ are the best positions for adding the cutting nodes. As a result, we can cut the original tree T and call Subroutine LessEqual (line 7 in More) to further reduce u_p into a leaf node. ■

Based on the above theorem and lemmas, we have the following theorem.

Theorem 2: The BUJI algorithm finds an optimal solution.

Proof: By Lemmas 2 and 3, lines 9–14 of Algorithm BUJI exhibit the greedy-choice property on subleaf nodes. Moreover, by Lemma 1, lines 2–8 of Algorithm BUJI also exhibit the greedy-choice property on leaf nodes. Therefore, Algorithm BUJI has the greedy-choice property on both leaf nodes and subleaf nodes. Since Algorithm BUJI has the greedy-choice property and the JITA problem has optimal substructure property (by Theorem 1), Algorithm BUJI finds an optimal cutting set, which is based on the theory presented in [3]. ■

V. Complexity Analysis

We analyze the time and space complexity of Algorithm BUJI in this section.

A. Time Complexity

In the BUJI algorithm, we use the bottom-up method to find the optimal solution for a given routing tree. We consider each leaf and each subleaf only once. Since every node in the tree might be a subleaf and might be cut into a leaf, we traverse each node at most twice. When we traverse a leaf node, it takes only constant time. When we traverse subleaf nodes using Subroutine LessEqual, it also takes only constant time. In Subroutine More, we use the linear-time SPLIT algorithm to find the set S_h. This step needs constant time on each node. Furthermore, we use constant time to cut the tree. Thus, the More subroutine requires constant time on each node. To sum up, we traverse each node at most twice, and in each traversal, we compute each node in constant time. Therefore, the total time complexity of the BUJI algorithm is $O(V)$, where V is the number of nodes in the given routing tree.

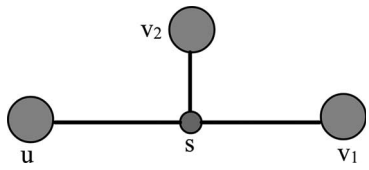

Fig. 11. u, v_1, and v_2 are gate terminals, and s is a Steiner point. Charges accumulated on edges $e(u, s)$, $e(s, v_1)$, and $e(s, v_2)$ will all cause antenna effect on u.

B. Space Complexity

All we need to store are the tree T and the cutting set C. A tree needs only $O(V + E)$ space. Moreover, according to the algorithm, we add at most two cutting nodes for each edge. Therefore, we need $O(E)$ space to keep the set C. Since $O(E) = O(V)$ in a tree, the total space complexity is $O(V)$. We have the following theorem.

Theorem 3: Algorithm BUJI optimally solves the JITA problem in $O(V)$ time using $O(V)$ space, where V is the number of vertices in the routing tree.

VI. Extensions

We extend the aforementioned optimal jumper-insertion algorithm on spanning trees to Stenier trees and consider the restrictions induced by routing obstacles in this section.

A. Jumper Insertion on Steiner Trees

Because Steiner points are just wire junctions, they cannot help discharge the wire. See Fig. 11 for an illustration, where u, v_1, and v_2 are gate terminals and s is a Steiner point. The charges accumulated on edges $e(u, s)$, $e(s, v_1)$, and $e(s, v_2)$ will all cause antenna effect on the gate terminal u.

As a result, we have to modify our algorithm to deal with Steiner trees. We present in this section an $O(V)$-time optimal algorithm, named Bottom-Up Jumper Insertion on Steiner Trees (BUJIST), for finding the minimum cutting set C for a given routing tree $T = (V, E)$ with V nodes. Let $T = (V, E)$ be a Steiner tree. Let the set V_G of nodes represent all gate terminals, and the set V_N of nodes represent all other nodes in the tree, including Steiner points, source nodes, and cutting nodes. Thus, we have $V = V_G \cup V_N$. Let the set E of edges denote the wires connecting the circuit terminals or junctions and an edge weight give the measure of the wires with the same

Algorithm: $BUJIST(T, L_{max}, t_{total}, C)$
Input: $T = (V = V_G \cup V_N, E)$ /* The given Steiner tree. */
L_{max} /* Upper Bound on antenna */
C /* Cutting set */
t_{total} /* Total Edge Length in T */
1 **for** each node $u \in V$
2 $w(u) \leftarrow 0$;
3 **while** $(t_{total} > L_{max})$
4 **for** each leaf node $u \in T$ not having been processed
5 Mark u as *processed*;
6 **if** $l(u, p(u)) + w(u) > L_{max}$
7 **if** $u \in V_G$
Let c be the node between u and $p(u)$ with
with $w(u) + l(u, c) = L_{max}$;
8 $C \leftarrow C \cup \{c\}$;
9 $t_{total} \leftarrow t_{total} - L_{max}$;
10 $T(V, E) \leftarrow T((V_G \setminus \{u\}) \cup (V_N \cup \{c\}), E \setminus \{e(u, c)\})$;
11 **for** each subleaf node $u_p \in T$
Let $u_1, u_2, \ldots, u_k$ denote all children nodes of u_p;
12 $totallen \leftarrow \sum_{i=1}^{k} (l(u_p, u_i) + w(u_i))$;
13 **if** u_p and all of its children are in V_N
14 $w(u_p) \leftarrow totallen$;
15 $T(V, E) \leftarrow T(V \setminus \cup_{i=1}^{k}\{u_i\}, E \setminus \cup_{i=1}^{k}\{e(u_i, u_p)\})$;
16 **else if** $totallen \leq L_{max}$
17 $LessEqualST(T, C, u_p, totallen)$;
18 **else**
19 $MoreST(T, C, u_p, totallen)$;

Fig. 12. Algorithm BUJIST deals with the leaf nodes first, and then calls Subroutines LessEqualST and MoreST to deal with the subleaf nodes.

unit as $L_{\max}$. For every node $u \in V$, let $w(u)$ be the weight function of u. The weight function records the accumulated edge weights that connect to node u from the processed nodes (Algorithm BUJIST is summarized in Fig. 12).

Now, we explain how to modify the BUJI algorithm for the BUJIST one.

Step 1) (lines 1–2 of Algorithm BUJIST): Set the initial weight function value for each node u. $w(u)$ records the edge weight that affects node u from the processed nodes. We first set $w(u) \leftarrow 0$ for every node $u \in V$, since no nodes have been processed in the beginning.

Step 2) (lines 4–10 of Algorithm BUJIST): Deal with every leaf node. Since we use $w(u)$ to record the accumulated edge weight from the processed nodes, we must use $w(u) + l(u, p(u))$ [instead of $l(u, p(u))$] to check whether node u satisfies the antenna rules or not. The other processes in this part remain the same.

Step 3) (lines 11–19 of BUJIST): Deal with every subleaf node. We have three possible cases to consider for this step. Let $\text{totallen} = \sum_{i=1}^{k} l(u_i, u_p) + w(u_i)$.

Case 3.1) u_p and all of its children are $\in V_N$In this case, all of u_p's children have been processed. Moreover, u_p and all its children are in V_N, and thus, they need not satisfy the antenna rule. Therefore, we just combine u_p and its children into a new leaf node and record totallen in $w(u_p)$ (see lines 13–15 of Algorithm BUJIST).

Case 3.2) $totallen \leq L_{\max}$: For this case, we apply the LessEqualST subroutine; see Fig. 13. If

Subroutine: LessEqualST(T, t_{total}, C, u_p, $totallen$, L_{max})
1 **if** $p(u_p)$ does not exist
2 **return**
3 **if** $totallen + l(u_p, p(u_p)) \leq L_{max}$
4 **if** $u_p \in V_N$
5 $w(u_p) \leftarrow totallen$;
6 **if** u_p or any of its children is in V_G
7 $V_N \leftarrow V_N \setminus \{u_p\}$;
8 $V_G \leftarrow V_G \cup \{u_p\}$ and mark u_p as unprocessed;
9 $T(V, E) \leftarrow T(V \setminus \cup_{i=1}^{k}\{u_i\}, E \setminus \cup_{i=1}^{k}\{e(u_i, u_p)\})$;
10 **else**
Let c be the node on $e(u_p, p(u_p))$
with $l(c, u_p) + totallen = L_{max}$;
11 $C \leftarrow C \cup \{c\}$;
12 $t_{total} \leftarrow t_{total} - L_{max}$;
13 $T(V, E) \leftarrow T(V \setminus (\cup_{i=1}^{k}\{u_i\}) \cup \{c\} \setminus \{u_p\},$
$E \setminus (\cup_{i=1}^{k}\{e(u_i, u_p)\}) \setminus \{e(u_p, c)\})$;

Fig. 13. Case when $\text{totallen} \leq L_{\max}$.

Subroutine: MoreST(T, t_{total}, C, u_p, $totallen$, L_{max})
1 $S = \cup_{i=1}^{k}\{l(e(u_i, u_p)) + w(u_i)\}$;
2 $S_h = SPLIT(S, L_{max})$;
Let c_i be the nodes between u_p and u_i with $l(c_i, u_p) = 0$,
$l(c_i, u_i) = l(u_p, u_i)$ and $\{l(e(u_i, u_p)) + w(u_i)\} \in S_h \; \forall 1 \leq i \leq |S_h|$;
3 $C \leftarrow C \cup \{c_i\} \; \forall \, 1 \leq i \leq |S_h|$;
4 $T(V, E) \leftarrow T(V \setminus \cup_{i=1}^{|S_h|}\{u_i\}, E \setminus \cup_{i=1}^{|S_h|}\{e(u_i, u_p)\})$;
5 $minuslen \leftarrow \sum_{s \in S_h} s$;
6 $t_{total} \leftarrow t_{total} - minuslen$;
7 $LessEqualST(T, t_{total}, C, u_p, totallen - minuslen, L_{max})$;

Fig. 14. Case when $\text{totallen} > L_{\max}$.

$\text{totallen} + w(u_p) + l(u_p, p(u_p)) \leq L_{\max}$, u_p will not violate the antenna rule. If $u_p \in V_N$, it must be a Steiner point, and all the edges between u_p and its children contribute to its weight. Thus, we simply combine u_p and its children into a new leaf node and update its weight as totallen (see lines 4–5 of Subroutine LessEqualST). Moreover, if u_p or any of its children is in V_G, it means that the new leaf node u_p has to satisfy the antenna rule, and thus, we add u_p into V_G. Otherwise, we let u_p be its original type (see lines 6–8 of Subroutine LessEqualST). The other processes in this part remain the same.

Case 3.3) $\text{totallen} > L_{\max}$: For this case, we apply the MoreST subroutine; see Fig. 14. Since we use $w(u)$ to record the accumulated edge weights from the processed nodes, we must use $w(u) + l(u, p(u))$ [instead of $l(u, p(u))$] to check whether node u satisfies the antenna rules or not. The other processes in this part remain the same.

In fact, the underlying ideas of Algorithm BUJIST are the same as those of Algorithm BUJI. Therefore, the optimality proof of Algorithm BUJIST for the JITA problem on Steiner

trees is similar to that of the BUJI algorithm. Moreover, all of the modifications in the BUJIST algorithm can be completed in constant times. Thus, the time complexity of the BUJIST algorithm is still $O(V)$. The additional space requirement of the BUJIST algorithm is the weight function $w(u)$ of every node $u \in V$, which uses $O(V)$ space. Therefore, the space complexity of Algorithm BUJIST is $O(V)$.

Theorem 4: Algorithm BUJIST optimally solves the JITA problem on Steiner trees in $O(V)$ time using $O(V)$ space, where V is the number of vertices in the Steiner tree.

B. Jumper Insertion on Steiner Trees With Obstacles

Since jumper-insertion routes a signal wire to the top-most layer, we must further consider the routing with obstacles in the active layers—the layers from the current routing layer up to the top-most layer, which could be prerouted nets, power/ground nets, clock nets, etc. We can modify the BUJIST algorithm to deal with the obstacles, called BUJIST with obstacles (BUJISTO). When some node u violates the antenna rules, we have to add a cutting node c on edge $e(u, p(u))$. However, if the position that the BUJIST algorithm wants to add a jumper has an obstacle in some upper layer, we can find the optimal substitution of c to evade from the obstacle. According to the BUJIST algorithm, we always add cutting nodes at critical positions on edges. Thus, if we find any substitution that is closer to $p(u)$ than c, node u will violate the antenna rules. As a result, we have to find a substitution from edge $e(u, p(u))$ that is closer to node u than the cutting node c. It is obvious that when we move a cutting node on edge $e(u, p(u))$ from node c toward node u, the first position that evades all obstacles is the best substitution of the cutting node c. We can use this method repeatedly when the selected positions of cutting nodes are occupied by obstacles.

VII. Experimental Results

We implemented the BUJIST, the BUJI, and the BUJISTO algorithms in the C++ language on a 1.6-GHz Intel Pentium PC with 256-MB memory under the Windows XP operating system. We performed two sets of experiments to verify the quality of 1) the BUJIST and the BUJI algorithms on randomly generated Steiner/spanning trees and 2) the BUJISTO algorithm on the layouts from the multilevel router [1], [7] on a set of commonly used MCNC benchmarks.

A. Randomly Generated Routing Trees

For the JITA problem, we compared this paper with the International Symposium on Physical Design (ISPD)-04 work [4] and four heuristic methods. The four heuristics are described as follows.

1) Heuristic Decreasing Degree with Decreasing Edge Length (**DDDE**):
   a) Sort the nodes with decreasing order by degrees and, then, process each node by this order.
   b) When dealing with a node u, we first sort its incident edges by decreasing edge lengths and then add jumpers on the edges by this order until all antenna violations are fixed. We apply the following jumper-insertion rule: If $L(u) \leq L_{\max}$, we need no jumpers for u to satisfy the antenna rule. Otherwise, we insert jumpers to satisfy the rule. Considering an edge e of u, if $L(u) - l(e) > L_{\max}$, we add a jumper on e just beside u to prevent the antenna violation caused by this edge. Therefore, $L(u)$ is reduced by the amount $l(e)$. Otherwise, if $L(u) - l(e) \leq L_{\max}$, we add a jumper on e such that $L(u)$ equals $L_{\max}$. With this insertion scheme, we can make sure that each node in the routing tree satisfies the antenna rule.
2) Heuristic Increasing Degree with Decreasing Edge Length (**IDDE**): This heuristic is the same as the DDDE heuristic, except that the nodes are sorted with increasing degrees in the first step.
3) Heuristic Decreasing Degree with Increasing Edge Length (**DDIE**): This heuristic is the same as the DDDE heuristic, except that the edges were sorted with increasing edge lengths in the second step.
4) Heuristic Increasing Degree with Increasing Edge Length (**IDIE**): This heuristic is the same as the DDDE heuristic, except that the nodes were sorted with increasing degrees in the first step, and the edges were sorted with increasing edge lengths in the second step.

For comparative study, we first randomly generated tree nodes on grid planes of the dimension 10^4 μm $\times$ 10^4 μm, assuming that each node is a gate terminal. Then, we constructed a Steiner tree and a minimum spanning tree based on the given gate terminals separately. We performed the following three experiments for our BUJIST algorithm, our BUJI algorithm, the ISPD-04 work [4], and the aforementioned four heuristics.

1) First, we apply the BUJIST algorithm on Steiner trees, the BUJI algorithm, the ISPD-04 method, and the four heuristics on the minimum spanning trees to find the number of jumpers required for each method to fix all antenna violations.
2) Second, for a given spanning tree, we find the minimum number of jumpers required for fixing all antenna violations for various $L_{\max}$ values.
3) Third, giving $L_{\max}$ as a constant, we find the running times for the algorithms and the heuristics to fix all antenna violations for routing trees with various numbers of nodes.

Table I shows the numbers of jumpers required for each method to fix all antenna violations for a routing tree with 10 to 250 gate terminals by changing $L_{\max}$ from 50 to 200 μm. Note that this $L_{\max}$ range is typical for the 90- to 250-nm CMOS technologies. Column 1 gives the number of gate terminals in the routing tree. Column 2 gives the $L_{\max}$ value, and Columns 3, 4, 6, 8, 10, 12, and 14 list the numbers of jumpers required (#J) for fixing the antenna violations for each $L_{\max}$ for the BUJIST algorithm on the Steiner trees, the BUJI algorithm, the ISPD-04 work [4], the DDDE, IDDE, DDIE, and IDIE heuristics on minimum spanning trees, respectively. Columns 5, 7, 9, 11, 13, and 15

TABLE I
COMPARISONS OF THE NUMBERS OF JUMPERS REQUIRED FOR BUJIST ON STEINER TREES, BUJI, ISPD-04, DDDE, IDDE, DDIE, AND IDIE ON MINIMUM SPANNING; TREES FOR FIXING ALL ANTENNA VIOLATIONS (NA: THE PERCENTAGE IS NOT AVAILABLE)

$node$ $number$	L_{max} (um)	BUJIST	BUJI		ISPD04		DDDE		IDDE		DDIE		IDIE	
		$\#J$	$\#J$	$\%More$	$\#J$	$\%More$	$\#J$	$\%More$	$\#J$	$\%More$	$\#J$	$\%More$	$\#J$	$\%More$
10	50	3	4	+ 33%	6	+ 100%	6	+ 100%	6	+ 100%	7	+ 133%	6	+ 100%
	100	0	1	NA	1	NA	1	NA	1	NA	1	NA	1	NA
	150	0	0	NA	0	NA	0	NA	0	NA	0	NA	0	NA
	200	0	0	NA	0	NA	0	NA	0	NA	0	NA	0	NA
30	50	22	28	+ 27%	34	+ 55%	31	+ 41%	30	+ 36%	39	+ 77%	37	+ 68%
	100	6	10	+ 67%	14	+ 133%	11	+ 83%	11	+ 83%	16	+ 167%	15	+ 150%
	150	1	4	+ 300%	4	+ 300%	4	+ 300%	4	+ 300%	4	+ 300%	5	+ 400%
	200	0	2	NA	2	NA	2	NA	2	NA	2	NA	2	NA
50	50	48	52	+ 8%	69	+ 44%	59	+ 23%	55	+ 15%	78	+ 63%	83	+ 73%
	100	20	27	+ 35%	36	+ 80%	32	+ 60%	32	+ 60%	38	+ 90%	39	+ 95%
	150	5	12	+ 140%	18	+ 260%	14	+ 180%	12	+ 140%	19	+ 280%	21	+ 320%
	200	1	6	+ 500%	6	+ 500%	6	+ 500%	6	+ 500%	8	+ 700%	8	+ 700%
100	50	120	135	+ 13%	160	+ 33%	140	+ 17%	136	+ 13%	180	+ 50%	178	+ 48%
	100	67	76	+ 13%	98	+ 46%	89	+ 33%	82	+ 22%	112	+ 67%	111	+ 66%
	150	32	47	+ 47%	68	+ 113%	54	+ 69%	54	+ 69%	71	+ 122%	69	+ 116%
	200	13	28	+ 115%	37	+ 185%	31	+ 138%	29	+ 123%	47	+ 262%	45	+ 246%
150	50	160	182	+ 14%	227	+ 42%	192	+ 20%	187	+ 17%	255	+ 59%	258	+ 61%
	100	68	92	+ 35%	125	+ 84%	107	+ 57%	101	+ 49%	136	+ 100%	129	+ 90%
	150	20	46	+ 130%	70	+ 250%	50	+ 150%	53	+ 165%	75	+ 275%	73	+ 265%
	200	4	16	+ 300%	19	+ 375%	17	+ 325%	18	+ 350%	24	+ 500%	24	+ 500%
200	50	191	214	+ 12%	272	+ 42%	232	+ 21%	226	+ 18%	300	+ 57%	307	+ 61%
	100	71	99	+ 39%	139	+ 96%	111	+ 56%	106	+ 49%	146	+ 106%	150	+ 111%
	150	21	43	+ 105%	57	+ 171%	48	+ 129%	44	+ 110%	67	+ 219%	65	+ 210%
	200	2	20	+ 900%	22	+1000%	20	+ 900%	20	+ 900%	25	+1150%	24	+1100%
250	50	205	245	+ 20%	309	+ 51%	264	+ 29%	255	+ 24%	348	+ 70%	355	+ 73%
	100	66	104	+ 58%	146	+ 121%	124	+ 88%	117	+ 77%	161	+ 144%	167	+ 153%
	150	12	44	+ 267%	56	+ 367%	47	+ 292%	46	+ 283%	72	+ 500%	65	+ 442%
	200	1	16	+1500%	17	+1600%	17	+1600%	16	+1500%	21	+2000%	21	+2000%

TABLE II
COMPARISONS OF THE NUMBERS OF JUMPERS REQUIRED FOR BUJI, ISPD-04, DDDE, IDDE, DDIE, AND IDIE FOR FIXING ALL ANTENNA VIOLATIONS BASED ON A ROUTING TREE OF 500 000 NODES

L_{max}	BUJI	ISPD04		DDDE		IDDE		DDIE		IDIE	
(um)	$\#Jump$	$\#Jump$	$\%More$	$\#Jump$	$\%More$	$\#Jump$	$\%More$	$\#Jump$	$\%More$	$\#Jump$	$\%More$
50	766367	904230	+18.0%	862462	+12.5%	862460	+12.5%	966272	+26.1%	966272	+26.1%
100	517851	729048	+40.8%	637065	+23.0%	637065	+23.0%	846602	+63.5%	846602	+63.5%
150	347545	518979	+49.3%	455950	+31.2%	455950	+31.2%	625962	+80.1%	625962	+80.1%
200	232063	336084	+44.8%	321626	+38.6%	321626	+38.6%	403554	+73.4%	403554	+73.4%
250	153694	213463	+38.9%	212490	+38.3%	212491	+38.3%	262731	+70.9%	262730	+70.9%
300	98465	129891	+31.9%	129834	+31.9%	129834	+31.9%	163218	+65.8%	163218	+65.8%
350	60270	74621	+23.8%	74619	+23.8%	74619	+23.8%	91773	+52.3%	91773	+52.3%
400	35178	41136	+16.9%	41136	+16.9%	41136	+16.9%	48017	+36.5%	48017	+36.5%
450	19522	21769	+11.5%	21769	+11.5%	21769	+11.5%	24340	+24.7%	24340	+24.7%
500	9873	10658	+7.95%	10658	+7.95%	10658	+7.95%	11808	+19.6%	11808	+19.6%
550	4501	4715	+4.75%	4715	+4.75%	4715	+4.75%	5191	+15.3%	5191	+15.3%
600	1925	1978	+2.75%	1978	+2.75%	1978	+2.75%	2141	+11.2%	2141	+11.2%
650	827	840	+1.57%	840	+1.57%	840	+1.57%	894	+8.10%	894	+8.10%
700	319	321	+0.63%	321	+0.63%	321	+0.63%	339	+6.27%	339	+6.27%
750	117	117	0.00%	117	0.00%	117	0.00%	120	+2.56%	120	+2.56%
800	40	40	0.00%	40	0.00%	40	0.00%	40	0.00%	40	0.00%

give the percentages of additional jumpers required $(\%\mathrm{More})$ for the respective BUJI, ISPD-04, DDDE, IDDE, DDIE, and IDIE methods over the BUJIST algorithm to fix all antenna violations, i.e., $\%\mathrm{More} = (\#\text{Jumpers of the method} - \#\text{Jumpers of BUJIST})/\#\text{Jumpers of BUJIST}$. However, if the BUJIST algorithm inserts no jumpers in the test case, we simply mark an "NA" in each corresponding field.

It is obvious that our BUJI and BUJIST algorithms outperform other methods significantly. Note that we list the results of the BUJIST algorithm here just for readers' reference. The BUJIST algorithm works on Steiner trees, while other methods all apply on minimum spanning trees. Since Steiner trees have less total edge lengths than minimum spanning trees, the accumulated charges on Steiner trees are also smaller than those on

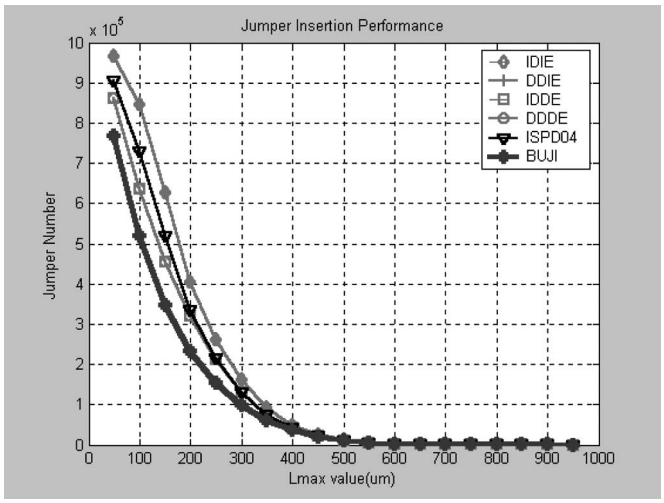

Fig. 15. Numbers of jumpers required to fix antenna violations for various $L_{\max}$ values.

minimum spanning trees. Therefore, the BUJIST algorithm gets some advantage over other methods in the number of jumpers required to fix the antenna violations. Nevertheless, judging from the results of the BUJI algorithm (which also works on minimum spanning trees), the behavior of our algorithm is the major key to the significant reduction in the jumpers required.

In order to have a fair comparison, we also compared the BUJI algorithm, the ISPD-04 method, and the four heuristics on the same minimum spanning trees. Moreover, in order to show the scalability of the algorithms/heuristics, we conducted the experiment on a minimum spanning tree with 500 000 nodes. Table II shows the numbers of jumpers required for fixing all antenna violations for the minimum spanning tree by changing $L_{\max}$ from 50 to 800 μm. Column 1 gives the $L_{\max}$ value, and Columns 2, 3, 5, 7, 9, and 11 list the numbers of jumpers required (#Jump) for fixing the antenna violations for each $L_{\max}$ for the BUJI algorithm, the ISPD-04 work [4], the DDDE, IDDE, DDIE, and IDIE heuristics, respectively. Columns 4, 6, 8, 10, and 12 give the percentages of additional jumpers required (%More) for the respective ISPD-04, DDDE, IDDE, DDIE, and IDIE methods over the BUJI algorithm to fix all antenna violations, i.e., $\%\text{More} = (\#\text{Jumpers of the heuristic} - \#\text{Jumpers of BUJI})/\#\text{Jumpers of BUJI}$. In Fig. 15, the numbers of jumpers required for fixing antenna violations are plotted as functions of the $L_{\max}$ values.
From the above experiments, tables, and curves, we have the following findings.

1) It is not surprising that BUJI performs much better than the other five methods. The phenomenon can be explained as follows: When we deal with nodes in a bottom-up manner, BUJI always pushes the jumper upward until at a position that just satisfies the antenna rule, adding more freedom and thus reducing the chance of antenna violations for the upper nodes. Therefore, BUJI can save a significant number of jumpers for antenna avoidance/fixing. In contrast, the other five methods do not have such an optimization scheme. Considering %More, BUJI outperforms ISPD-04 by as large as 49.3% for $L_{\max} = 150$ μm, DDDE and IDDE by as large as 38.6% for $L_{\max} = 200$ μm, and DDIE and IDIE by as large as 80.1% for $L_{\max} = 150$ μm. Considering the number of jumpers inserted, BUJI requires fewer jumpers than the other five methods. Moreover, the smaller the $L_{\max}$, the bigger the difference in the number of jumpers required. For example, when $L_{\max} = 50$ μm, ISPD-04 needs 137 863 more jumpers than BUJI, the DDDE and IDDE heuristics need 96 095 more jumpers, and the DDIE and IDIE heuristics even need 199 905 more jumpers. However, when $L_{\max} = 800$ μm, the results for the six methods are the same. This is a reasonable phenomenon. The smaller the $L_{\max}$, the more likely the antenna violation occurs. Therefore, more jumpers are needed, and more decisions of choosing the positions for the jumpers must be made. Thus, the superiority of BUJI is more apparent. When $L_{\max}$ is larger, fewer antenna violations occur and thus fewer jumpers are needed for fixing the violations. Moreover, the ranges for feasible jumper insertions are larger because the tolerances for the jumper positions are larger.
2) Heuristics DDDE and IDDE obtain almost the same results and so do the heuristics DDIE and IDIE. Furthermore, heuristics DDDE and IDDE outperform the heuristics DDIE and IDIE. The results reveal that edge lengths are more critical than node degrees for jumper insertion for fixing antenna violations. Moreover, the result shows that adding jumpers on edges in a decreasing edge-length order results in better solution quality than adding jumpers in an increasing edge-length order. This phenomenon is consistent with the underlying idea used in the More subroutine of the BUJI algorithm.

Table III shows the CPU times required for antenna fixing on routing trees of the number of nodes ranging from 100 000 to 1 000 000, based on $L_{\max} = 50$ μm. Column 1 gives the numbers of nodes in the routing trees. Because the numbers of nodes are so huge, the program spent most of the CPU times on reading the input files. Therefore, we list the CPU times for reading the input files and running the algorithm/heuristics separately in order to examine the time complexity more closely. The second column (File) gives the CPU times for reading the input files. The first (Main) and second (Total) columns in each algorithm/heuristic give the respective CPU times for executing the main body of the algorithm/heuristic and running the both parts combined.

In Fig. 16, the CPU time for executing the main body of the algorithm/heuristic is plotted as a function of the number of nodes in the routing tree. As shown in the table and the figure, the empirical running times for the six methods are close to linear, and BUJI requires slightly smaller CPU times than the other five methods for fixing antenna violations. In particular, BUJI requires only 6.5 s to find an optimal solution for a routing tree of one million nodes. Therefore, the BUJI algorithm can handle a test case of a very huge number of nodes in very short time. Fig. 17 shows a layout resulting from BUJI with 429 jumpers for antenna fixing on a routing tree with 1615 nodes on the plane, based on $L_{\max} = 500$ μm.

TABLE III
COMPARISONS OF THE CPU TIMES REQUIRED FOR BUJI, ISPD04, DDDE, IDDE, DDIE, AND IDIE TO FIX THE ANTENNA VIOLATIONS, BASED ON $L_{\max} = 50\ \mu$m

node number	*File*	BUJI		ISPD04		DDDE		IDDE		DDIE		IDIE	
	Time(sec)	*Time(sec)*		*Time(sec)*		*Time(sec)*		*Time(sec)*		*Time(sec)*		*Time(sec)*	
	File	*Main*	*Total*	*Main*	*Total*	*Main*	*Total*	*Main*	*Total*	*Main*	*Total*	*Main*	*Total*
100000	1.578	0.438	2.016	0.422	2.000	0.468	2.046	0.453	2.031	0.453	2.031	0.453	2.031
200000	3.125	1.016	4.141	0.953	4.078	1.015	4.140	1.000	4.125	1.000	4.125	1.047	4.172
300000	4.766	1.547	6.313	1.532	6.298	1.610	6.376	1.594	6.360	1.610	6.376	1.594	6.360
400000	6.282	2.156	8.438	2.156	8.438	2.140	8.422	2.172	8.454	2.172	8.454	2.172	8.454
500000	7.969	2.813	10.782	2.782	10.751	2.891	10.860	2.953	10.922	2.891	10.860	2.891	10.860
600000	9.547	3.484	13.031	3.500	13.047	3.578	13.125	3.609	13.156	3.469	13.016	3.688	13.235
700000	11.141	4.172	15.313	4.157	15.298	4.187	15.328	4.281	15.422	4.313	15.454	4.281	15.422
800000	12.765	4.891	17.656	4.891	17.656	5.062	17.827	4.937	17.702	4.968	17.733	5.125	17.890
900000	14.344	5.688	20.032	5.578	19.922	5.875	20.219	5.843	20.187	5.688	20.032	5.688	20.032
1000000	16.015	6.453	22.468	6.453	22.468	6.453	22.468	6.609	22.624	6.563	22.578	6.593	22.608

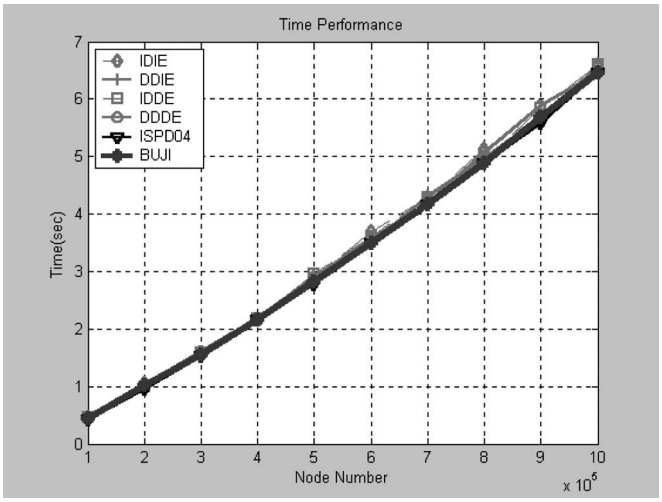

Fig. 16. CPU times required for antenna fixing for routing trees of various numbers of nodes.

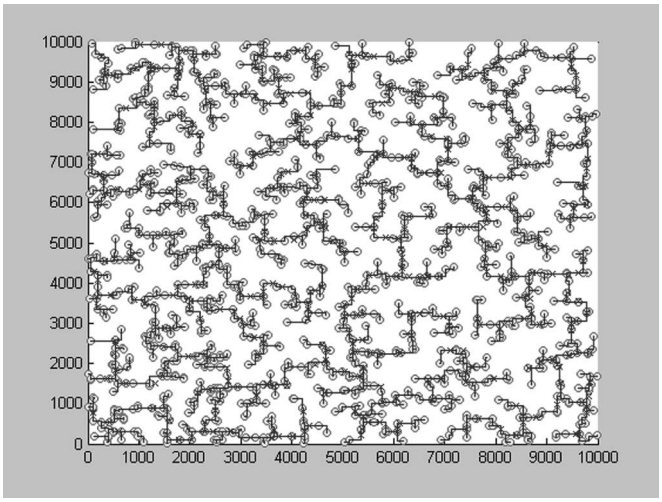

Fig. 17. Layout resulting from the BUJI algorithm with 429 jumpers for antenna fixing on a tree with 1615 nodes. Circles denote the nodes of the routing tree and the $\times$ signs denote the inserted jumpers.

B. MCNC Layouts

We also performed experiments based on the layouts generated from the multilevel routers in the study in [1] and [7] on a set of commonly used MCNC benchmarks to test the quality of our BUJISTO algorithm. The statistics of the benchmark circuits are listed in Table IV. Six test cases are chosen from the MCNC benchmarks since only these test cases record the source and sink information for each net. The column "Circuit" denotes the circuit name, "Size" denotes the circuit dimension, "# Layers" denotes the number of routing layers, "# Nets" denotes the number of nets, and "# Pins" denotes the number of pins.

TABLE IV
MCNC BENCHMARK STATISTICS

Circuit	Size (μm^2)	# Layers	# Nets	# Pins
s5378	435×239	3	1693	4818
s9234	404×225	3	1476	4260
s13207	660×365	3	3777	10776
s15850	705×389	3	4470	12793
s38417	1144×619	3	11308	32344
s38584	1295×672	3	14753	42931

TABLE V
ANTENNA FIXING RATES

Circuit Name	$L_{max} = 50\mu m$			$L_{max} = 100\mu m$		
	Total # Viol.	# Fixed Viol.	Fixing Rate (%)	Total # Viol.	# Fixed Viol.	Fixing Rate (%)
s5378	95	65	68.42	49	44	89.80
s9234	56	34	60.71	22	17	77.27
s13207	164	86	52.44	83	51	61.45
s15850	182	93	51.10	98	54	55.10
s38417	406	231	56.90	184	122	66.30
s38584	550	341	62.00	283	167	59.01
	Avg.		58.60	Avg.		68.16

The antenna threshold $L_{\max}$ set in [4] is 100 μm; in this paper, 50 and 100 μm are both tested. See Table V for the experimental results. The results show that our BUJISTO algorithm can significantly reduce the number of antenna violations. For $L_{\max} = 100\ \mu$m, the fixing rates range from 55.10% to 89.80% with the average 68.16%; for $L_{\max} = 50\ \mu$m, the fixing rates range from 55.10% to 68.4% with the average 58.60%. The fixing rate for $L_{\max} = 100\ \mu$m is higher, since a larger antenna threshold leads to higher flexibility for finding a cutting point for jumper insertion. Furthermore, the fixing rates highly depend on the routing configurations, particularly the routing obstacles in the current and upper layers for jumper

insertion. A sparser layout implies fewer obstacles for jumper insertion. Consequently, the sparser the layout, the higher the fixing rate.

VIII. Conclusion

We have presented a linear-time optimal jumper-insertion algorithm for avoiding/fixing antenna violations on routing trees. It is the first optimal algorithm for the general tree-cutting problem. Empirical results have shown that our algorithm uses linear time and obtains solutions of very high quality. This paper can apply to any routing trees (could be a net to be globally routed or a net after detailed routing) and thus readily be incorporated into a router for antenna-effect avoidance or a postlayout optimizer for antenna-violation fixing.

Acknowledgment

The authors would like to thank Z.-W. Jiang for providing the results on the MCNC layouts.

References

[1] Y.-W. Chang and S.-P. Lin, "MR: A new framework for multilevel full-chip routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 5, pp. 793–800, May 2004.

[2] P. H. Chen, S. Malkani, C.-M. Peng, and J. Lin, "Fixing antenna problem by dynamic diode dropping and jumper insertion," in *Proc. IEEE Int. Symp. Quality Electron. Des.*, 2000, pp. 275–282.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. New York: McGraw-Hill, 2001.

[4] T.-Y. Ho, Y.-W. Chang, and S.-J. Chen, "Multilevel routing with antenna avoidance," in *Proc. ACM Int. Symp. Phys. Des.*, Apr. 2004, pp. 34–40.

[5] S. Krishnan *et al.*, "Assessment of charge-induced damage to ultra-thin gate MOSFETs," in *IEDM Tech. Dig.*, 1997, pp. 445–448.

[6] S. Kundu and J. Misra, "A linear tree partitioning algorithm," *SIAM J. Comput.*, vol. 6, no. 1, pp. 151–154, Mar. 1977.

[7] S.-P. Lin and Y.-W. Chang, "A novel framework for multilevel routing considering routability and performance," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 2002, pp. 44–50.

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

[9] H. Watanabe *et al.*, "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.

[10] D. Wu, J. Hu, and R. Mahapatra, "Coupling aware timing optimization and antenna avoidance in layer assignment," in *Proc. ACM Int. Symp. Phys. Des.*, Apr. 2005, pp. 20–27.

Bor-Yiing Su received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2005. He plans to conduct graduate study in the Department of Electrical Engineering at the topmost universities in America and to obtain a Ph.D. degree in the area of physical design.

He is currently working at SpringSoft Company, Hsinchu, Taiwan, R.O.C., where he is currently working on large-scale congestion-driven placement. His current research interest is placement-related topics.

Mr. Su was the recipient of the Presidential Award from the National Taiwan University for four semesters during his college years.

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

He is a Professor in the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University. He is currently also a Visiting Professor at Waseda University, Japan. He was with the IBM T. J. Watson Research Center, Yorktown Heights, NY, in the summer of 1994. From 1996 to 2001, he was on the faculty of National Chiao Tung University, Taiwan. His current research interests lie in VLSI physical design, design for manufacturing, and FPGA. He has been working closely with industry on projects in these areas. He has coauthored one book on routing and over 120 ACM/IEEE conference/journal papers in these areas.

Dr. Chang received an award at the 2006 ACM ISPD Placement Contest, Best Paper Award at ICCD-1995, and nine Best Paper Award Nominations from DAC-2007, ISPD-2007 (two), DAC-2005, 2004 ACM TODAES, ASP-DAC-2003, ICCAD-2002, ICCD-2001, and DAC-2000. He has received many awards for research performance, such as the inaugural First-Class Principal Investigator Awards and the 2004 Mr. Wu Ta You Memorial Award from the National Science Council of Taiwan, the 2004 MXIC Young Chair Professorship from the MXIC Corp, and for excellent teaching from National Taiwan University and National Chiao Tung University. He is an editor of the *Journal of Computer and Information Science*. He has served on the ACM/SIGDA Physical Design Technical Committee and the technical program committees of ASP-DAC (topic chair), DAC, DATE, FPT (program co-chair), GLSVLSI, ICCAD, ICCD, IECON (topic chair), ISPD, SOCC (topic chair), TENCON, and VLSI-DAT (topic chair). He is currently an independent board member of Genesys Logic, Inc, the chair of the Design Automation and Testing (DAT) Consortium of the Ministry of Education, Taiwan, a member of the board of governors of the Taiwan IC Design Society, and a member of the IEEE Circuits and Systems Society, ACM, and ACM/SIGDA.