An Optimal Jumper Insertion Algorithm for Antenna Avoidance/Fixing on General Routing Trees with Obstacles

Bor-Yiing Su and Yao-Wen Chang
National Taiwan University

Jiang Hu
Texas A&M University
The process antenna effect is a phenomenon of plasma-induced gate oxide degradation caused by charge accumulation on conductors. During metallization, each time an additional layer of interconnect is added. While the metal line is being manufactured, the floating interconnect acts as a temporary capacitor to store charges induced from plasma etching.

A two-pin net

- Metal 3
- Metal 2
- Metal 1
If the charged conductor is connected only to the gate oxide, Fowler-Nordheim (F-N) tunneling current will discharge through the thin oxide and damage the gate.

- No harm to diffusion source (discharge through substrate)
Solutions for Antenna Effect

- **Jumper Insertion**
  - Break the signal wires with antenna violations and route them to the top-most layer
  - Induce vias

- **Embedded Protection Diode**
  - Add a protection diode on every input port of a cell
  - May consume unnecessary areas

- **Diode Insertion**
  - Insert “under-the-wire” diodes to fix the violation
  - Need extra silicon space to place the diodes
Solutions for Antenna Effect

- **Jumper Insertion**
  - Break the signal wires with antenna violations and route them to the highest layer
  - Induce vias

- **Embedded Protection Diode**
  - Add a protection diode on every input port of a cell
  - May consume unnecessary areas

- **Diode Insertion**
  - Insert “under-the-wire” diodes to fix the violation
  - Need extra silicon space to place the diodes
Jumper Insertion

- Jumper insertion reduces the charge amount for violated nets during manufacturing
  - **Side effects**: Delay and congestion

A two-pin net

A two-pin net with jumper insertion
Steiner Tree

- Steiner points are just wire junctions. They cannot help discharge the wire.

- **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 $u$. 

![Diagram of Steiner Tree](https://via.placeholder.com/150)
Routing Obstacles

- Since a jumper routes a signal wire to the topmost layer, we must consider the routing with obstacles in the active layers, e.g., pre-routed nets, power/ground nets, clock nets.
  - Active layers: the layers from the current routing layer up to the topmost layer
- Need to avoid obstacles when adding jumpers

Invalid!

Diagram:

- u₁ and u₂ are endpoints.
- Jumper connects u₁ to u₂.
- Obstacle is between the jumper and the path.
Previous Work

- **Ho, Chang & Chen, ISPD-04**
  - Bottom-up dynamic programming in loglinear time
  - Insert jumpers right beside nodes of a spanning tree

- **Wu, Hu & Mahapatra, ISPD-05**
  - Insert jumpers at arbitrary positions of edges of a Steiner tree in linear time
  - Optimal only for some special tree topologies

- **Su and Chang, DAC-05**
  - Insert jumpers at arbitrary positions of edges of a spanning tree in loglinear time
  - Optimal for a spanning tree with arbitrary topologies
## Previous Work

Previous works are optimal only for restricted cases & do not consider obstacles!!

<table>
<thead>
<tr>
<th></th>
<th>ISPD-04</th>
<th>DAC-05</th>
<th>ISPD-05</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optimal for general routing tree?</strong></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Consider obstacles?</strong></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Consider Steiner trees?</strong></td>
<td>N</td>
<td>N</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Allow arbitrary insertion positions?</strong></td>
<td>N</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Outline

- Introduction to Antenna Effect
- Problem Definition
- An Optimal Algorithm for Jumper Insertion
- Complexity Analysis
- Experimental Results
- Conclusions
Formulate the problem of jumper insertion on a routing tree for antenna avoidance/fixing as a tree-cutting problem.

\[ T = (V_G \cup V_N, E) : \text{a Steiner tree} \]

- Set \( V_G \) of nodes represents all gate terminals.
- Set \( V_N \) of nodes represents other nodes.
- Set \( E \) of edges denotes the wires connecting the circuit terminals or junctions.

\[ D : \text{set of obstacles in the active layers} \]

Projection of the obstacles in \( D \) defines the forbidden regions for jumper insertion.

\[ F : \text{set of the forbidden regions.} \]

- \( f(u) = 1 \) if node \( u \) is in a forbidden region \((u \in F)\);
- \( f(u) = 0 \), otherwise.
The edge weight in a Steiner tree models the strength of antenna effect caused by the corresponding wire.

The edge weight can be

- the length of the wire
- the area of the wire
- the perimeter of the wire
- the ratio of antenna strength to gate size, or
- any other reasonable models.
Problem JIROA

Problem **JIROA** (*Jumper Insertion on a Routing tree with Obstacles for Antenna avoidance/fixing*)

- **Input:** A routing tree $T = (V_G \cup V_N, E)$, an upper bound $L_{max}$, and a set $D$ of rectangular obstacles

- **Goal:** find the minimum set $C$ of cutting nodes

- **Constraint:** $c \neq u$ for any $c \in C$ and $u \in V$, $f(c) = 0$, $\forall c \in C$, such that $L(u) \leq L_{max}$, $\forall u \in V_G$.

  - $L(u)$: sum of effective edge weights on node $u$
  - $L_{max}$: antenna upper bound
Outline

- Introduction to Antenna Effect
- Problem Definition
- An Optimal Algorithm for Jumper Insertion
- Complexity Analysis
- Experimental Results
- Conclusion
A **subleaf** is a node for which all its children are leaf nodes, and all the children have been processed.

Let \( u, v \) be two adjacent nodes with \( f(v) = 1 \). Then \( r(u, v) \) denotes the cutting node \( c \) on edge \( e = (u, v) \) with \( f(c) = 0 \) and \( l(u, c) \) (weight between nodes \( u \) and \( c \)) being the maximum among every node on edge \( e \).
Algorithm BUJIO (Bottom Up Jumper Insertion with Obstacles) for jumper insertion applies a bottom-up approach to insert cutting nodes on a routing tree.

- Step 1: Sort the obstacles in $D$ by the $x$-axes and then the $y$-axes
- Step 2: Compute the initial weight for every node
- Step 3: Makes every leaf node satisfy the antenna rule
- Step 4: Make every subleaf node satisfy the antenna rule, and then cut the corresponding leaf nodes to turn a subleaf node into a new leaf node
Algorithm BUJIO (Bottom Up Jumper Insertion with Obstacles) for jumper insertion uses a bottom-up approach to insert cutting nodes on a routing tree.

- **Step 1:** Sort the obstacles in $D$ by the $x$-axes and then the $y$-axes.
- **Step 2:** Compute the weight of every node.
- **Step 3:** Let every **leaf node** satisfy the antenna rule.
- **Step 4:** Make every **subleaf node** satisfy the antenna rule, and then cut the corresponding leaf nodes to turn a subleaf node into a new leaf node.

Iteratively apply Steps 3 & 4 until all gate terminals satisfy the antenna rule.
Steps 1 and 2: Initialization

- **Step 1:** Sort the obstacles in $D$ by the $x$-axes and then the $y$-axes
  - Determine $f(u)$ for some node $u$ in $O(\lg D)$ time

- **Step 2:** Compute the initial weight of every node
  - $w(u)$: accumulated edge weights on node $u$

\[ w(u) = l(u,d_1) + l(u,d_2) + l(u,d_3) \]
Step 3: Leaf Node Processing

- Step 3: Make every leaf node which is a gate terminal satisfy the antenna rule

\[
I(u, p(u)) + w(u) > L_{\text{max}}
\]

\[f(c) = 0\]

\[
L_{\text{max}} - w(u)
\]

Tree node

\[p(u) : \text{parent of } u\]

\[l(e), l(u, v) : \text{weight of the edge } e = (u, v)\]

\[
L_{\text{max}} : \text{upper bound on antenna}\]

\[C : \text{cutting set}\]
Step 4: Subleaf Node Processing

- **Step 4:** Make every **subleaf node** satisfy antenna rule
- **totallen**(uᵢ): total weights of the edges between the node uᵢ and its children

\[
totallen(u_p) = \sum_{i=1}^{k} l(u_i, u_p) + w(u_i)
\]

- **uᵢ** : a subleaf node
- **uᵢ** : subleaf’s children, \( \forall 1 \leq i \leq k \)

- **Classify the subleaf nodes according to totallen**
  - Case 1: uᵢ and all its children are not gate terminals
  - Case 2: \( totallen(u_p) + w(u_p) \leq L_{\text{max}} \)
  - Case 3: \( totallen(u_p) + w(u_p) > L_{\text{max}} \)
Case 1: $u_p$ and All Its Children Are in $V_N$

- $u_p$ and all its children are **not gate terminals**, so no antenna violation occurs.
  - $w(u_p) \leftarrow w(u_p) + \text{totallen}(u_p)$
  - Cut off $u_p$’s children to turn $u_p$ into a **leaf node**

$$w(u_p) \leftarrow w(u_p) + \text{totallen}(u_p)$$

Steiner point
Case 2: $\text{totallen}(u_p) + w(u_p) \leq L_{\text{max}}$

- If $u_p$ has a parent
  - If $\text{totallen}(u_p) + w(u_p) + I(u_p, p(u_p)) \leq L_{\text{max}}$
    - If $u_p \in V_N$, assign $w(u_p) \leftarrow w(u_p) + \text{totallen}(u_p)$
    - Cut $u_p$'s children from the tree, make $u_p$ a leaf node
  - Else insert the cutting node to make $\text{totallen}(u_p) + w(u_p) + I(u_p, c) = L_{\text{max}}$
    - If $f(c) = 1$, use $c_1 = r(u_p, c)$ to replace $c$
    - Reduce the tree to make $c$ or $c_1$ a leaf node

\[
w(u_p) \leftarrow w(u_p) + \text{totallen}(u_p) \quad \text{totallen}(u_p) + w(u_p) + I(u_p, c) = L_{\text{max}}
\]
Case 3: \( \text{totallen}(u_p) + w(u_p) > L_{\max} \)

- **Case 3:** \( \text{totallen}(u_p) + w(u_p) > L_{\max} \)
  - Step 1: \( S = \bigcup_{i=1}^{k} \{ l(e(u_i, u_p)) + w(u_i) \} \)
  - Step 2: Find \( S = S_l \cup S_h, S_l \cap S_h = \emptyset \), such that
    - For any \( a \in S_l \) and \( b \in S_h \), \( a \leq b \)
    - \( \sum_{s \in S} s \leq L_{\max} - w(u_p) \)
    - For any \( b \in S_h \), \( \sum_{s \in S} s + b > L_{\max} - w(u_p) \)
  - Step 3: Add cutting nodes \( c_1, \ldots, c_{S_h} \) on every \( s \in S_h \)
  - Step 4: Use **Case 2** to cut \( u_p \) into a leaf node
An Example
An Example

\[ w(u_5) = 5 \]
An Example

L_{\text{max}} = 10
An Example

L_{max} = 10

w(u_5) = 5
An Example

$L_{\text{max}} = 10$
An Example

$L_{\text{max}} = 10$
An Example

$\text{L}_{\text{max}} = 10$
An Example

Totally we insert 8 jumpers on this Steiner tree!
Outline

- Introduction to Antenna Effect
- Problem Definition
- An Optimal Algorithm for Jumper Insertion
- Complexity Analysis
- Experimental Results
- Conclusions
Time & Space Complexity

- The tree-cutting problem JIROA exhibits the properties of optimal substructures and greedy choices.

- Algorithm BUJIO optimally solves the JIROA problem in $O((V + D) \log D)$ time using $O(V)$ space
  - $V$: # of tree nodes
  - $D$: # of obstacles
Outline

- Introduction to Antenna Effect
- Problem Definition
- An Optimal Algorithm for Jumper Insertion
- Complexity Analysis
- Experimental Results
- Conclusions
Experimental Settings

- Implemented BUJIO in C++ on a 2.4 GHz Intel Pentium PC with 256 MB memory in Windows XP
- Generated gate terminals on 1 cm x 1 cm planes with a number of nodes and obstacles
- Compared with ISPD-04, ISPD-05, DAC-05 works, extending their works to handle obstacles using the same obstacle processing in this paper
- Our BUJIO and the ISPD-05 algorithms work on Steiner trees while the ISPD-04 and DAC-05 ones on spanning trees.
### Experiment #1: #Jumpers for Random Cases

- **Random cases** (10,000 nodes, 500 obstacles)
- **Quality rating:** BUJIO > ISPD-05 > DAC-05 > ISPD-04
- **Runtimes are about the same for all methods (0.15 sec)**

<table>
<thead>
<tr>
<th>$L_{\text{max}}$ (um)</th>
<th>BUJIO</th>
<th>ISPD-05</th>
<th>ISPD-04</th>
<th>DAC-05</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#J</td>
<td>#J</td>
<td>%More</td>
<td>#J</td>
</tr>
<tr>
<td>220</td>
<td>1533</td>
<td>1537</td>
<td>+0.3%</td>
<td>2359</td>
</tr>
<tr>
<td>230</td>
<td>1341</td>
<td>1348</td>
<td>+0.5%</td>
<td>2020</td>
</tr>
<tr>
<td>240</td>
<td>1144</td>
<td>1149</td>
<td>+0.4%</td>
<td>1742</td>
</tr>
<tr>
<td>250</td>
<td>961</td>
<td>966</td>
<td>+0.5%</td>
<td>1501</td>
</tr>
<tr>
<td>260</td>
<td>823</td>
<td>827</td>
<td>+0.5%</td>
<td>1279</td>
</tr>
<tr>
<td>270</td>
<td>706</td>
<td>708</td>
<td>+0.3%</td>
<td>1109</td>
</tr>
<tr>
<td>280</td>
<td>599</td>
<td>601</td>
<td>+0.2%</td>
<td>928</td>
</tr>
<tr>
<td>290</td>
<td>517</td>
<td>518</td>
<td>+0.2%</td>
<td>767</td>
</tr>
<tr>
<td>300</td>
<td>434</td>
<td>435</td>
<td>+0.2%</td>
<td>645</td>
</tr>
</tbody>
</table>

| %More                  | +0.3% | +52%   | +25%   |
Experiment #2: #Jumpers for Synthetic Cases

- Synthetic cases: make gate terminals adjacent to each other
- Quality rating: BUJIO > DAC-05 > ISPD-05 > ISPD-04

<table>
<thead>
<tr>
<th>$L_{\text{max}}$ (um)</th>
<th>BUJIO</th>
<th>ISPD-05</th>
<th>ISPD-04</th>
<th>DAC-05</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#J</td>
<td>#J</td>
<td>%More</td>
<td>#J</td>
</tr>
<tr>
<td>220</td>
<td>4272</td>
<td>5455</td>
<td>+27.6%</td>
<td>5943</td>
</tr>
<tr>
<td>230</td>
<td>4157</td>
<td>5311</td>
<td>+27.8%</td>
<td>5762</td>
</tr>
<tr>
<td>240</td>
<td>4056</td>
<td>5174</td>
<td>+27.6%</td>
<td>5604</td>
</tr>
<tr>
<td>250</td>
<td>3972</td>
<td>5061</td>
<td>+27.4%</td>
<td>5455</td>
</tr>
<tr>
<td>260</td>
<td>3889</td>
<td>4951</td>
<td>+27.3%</td>
<td>5299</td>
</tr>
<tr>
<td>270</td>
<td>3814</td>
<td>4844</td>
<td>+27.0%</td>
<td>5157</td>
</tr>
<tr>
<td>280</td>
<td>3728</td>
<td>4738</td>
<td>+27.1%</td>
<td>5028</td>
</tr>
<tr>
<td>290</td>
<td>3657</td>
<td>4637</td>
<td>+26.8%</td>
<td>4915</td>
</tr>
<tr>
<td>300</td>
<td>3605</td>
<td>4541</td>
<td>+26.0%</td>
<td>4796</td>
</tr>
</tbody>
</table>

+27%  +36%  +5%
Resulting Layout

- 1000 nodes, 500 obstacles, 426 jumpers, $L_{\text{max}} = 500$ um
Conclusions

- We have presented the first optimal algorithm to solve the JIROA problem in $O((V+D) \lg D)$ time and $O(V)$ space, where $V$ is the number of vertices and $D$ is the number of obstacles.

- Experimental results have shown that our algorithm leads to more robust and much better solutions than the previous works.

- Our work can be applied to any routing trees, and thus readily be incorporated into a router for antenna avoidance or a post-layout optimizer for antenna fixing.
Thank You!!
Obstacle Example

Collision! Cannot insert a jumper here!

Antenna violation occurs!

No collision occurred!

We can insert a jumper here.