A Local Optimal Method on DSA Guiding Template Assignment with Redundant/Dummy Via Insertion

Xingquan Li\textsuperscript{1}, Bei Yu\textsuperscript{2}, Jianli Chen\textsuperscript{1}, Wenxing Zhu\textsuperscript{1},

\textsuperscript{1}Fuzhou University
\textsuperscript{2}The Chinese University of Hong Kong

24th Asia and South Pacific Design Automation Conference
Tokyo Japan 2019
Outline

Introductions
Problem Formulation
Algorithm
Experimental Results
Conclusions
**Block Copolymers Directed Self-Assembly (DSA)**

- **Block copolymer (BCP)**
  - A unique string of two types of polymer.
  - One type of polymer is hydrophilic and another is hydrophobic.

- **Nanostructures**
  - Cylinders, spheres, and lamellae.
  - The cylindrical nanostructure is suitable for patterning contacts and vias.

![Diagram of Block Copolymers](image-url)
DSA Process

- Vias are not regularly placed in practical layout.
- A simple regular hole array generated by standard DSA is not suitable for IC fabrication.
- Topological template guided DSA process has been proposed to support patterning irregularly vias layout.
- Closed vias are grouped; And a guiding template is identified for each group.
Given a vias layout, we should assign the guiding templates for every via.

Guiding templates are patterned on a wafer through optical lithography.

Each guiding template is filled with BCPs.

DSA can be controlled by thermal annealing process.
Guiding Templates

- Pre-defined DSA pattern set to improve robust.
- Within-group contact/via distance.
- Complex shapes are difficult to print.
- Unexpected holes and placement error of holes for some patterns.
- The distance of any two guiding templates should larger than minimum optical resolution spacing $d_s$. 

![Diagram of guiding templates with contact/via distances and hole placements]
Redundant Via Insertion (RVI)

- Insert an extra via near a single via.
- Prevent via failure, improve circuit yield and reliability.
Due to the characteristic of DSA, vias in a group must match some specific patterns so that they can be assigned to the same guiding template.

Increase the choices to form guiding templates with the help of dummy via insertion.

Case 1

Case 2

Case 3

Case 4
Input

- Post-routing layout
- Usable DSA guiding templates
- Optical resolution limit space

Usable DSA guiding templates

Optical resolution limit space

Post-routing layout
DSA Guiding Template Assignment with Redundant/Dummy Via Insertion (DRDV)

- **Input**
  - Post-routing layout
  - Usable DSA guiding templates
  - Optical resolution limit space

- **Output**
  - Redundant via insertion for every via
  - Guiding template assignment with suitable dummy vias for every via and redundant via
DSA Guiding Template Assignment with Redundant/Dummy Via Insertion (DRDV)

- **Input**
  - Post-routing layout
  - Usable DSA guiding templates
  - Optical resolution limit space

- **Output**
  - Redundant via insertion for every via
  - Guiding template assignment with suitable dummy vias for every via and redundant via

- **Constraints**
  - Inserted redundant vias should be legal
  - The spacing between neighboring guiding template should larger than the optical resolution limit space

- **Objectives**
  - Maximize the number (ratio) of inserted redundant vias
  - Maximize the number (ratio) of patterned vias by DSA
Outline

- Introductions
- Problem Formulation
- Algorithm
- Experimental Results
- Conclusions
Solution Flow

Preprocessing
- Find All Redundant/Dummy Via Candidates
- Detect Building Blocks
- Construct Conflict Graph

Local Optimal Solver
- Integer Linear Programming Formulation
- Initial Solution Generation
- Unconstrained Nonlinear Programming Solver

Redundant/Dummy Via Insertion with Template Assignment

DSA Guiding Template
Routing Layout
Optical resolution limit spacing $d_s$
Preprocessing

- DSA Guiding Template
- Routing Layout
- Optical resolution limit spacing $d_s$

**Preprocessing**

- Find All Redundant/Dummy Via Candidates
- Detect Building Blocks
- Construct Conflict Graph

**Local Optimal Solver**

- Integer Linear Programming Formulation
- Initial Solution Generation
- Unconstrained Nonlinear Programming Solver

Redundant/Dummy Via Insertion with Template Assignment
Redundant/Dummy Via Candidates

- Redundant via candidate
  - It should be inserted next to every via.
  - It should not overlap with any metal wire from other nets of wires.

- Dummy via candidate
  - It can make up a multi-hole (not less than three holes) guiding template with other vias or redundant vias.
  - It can improve the insertion rate or manufacture rate.

- Find all redundant/dummy via candidates for every via in time $O(n)$.
Building-Blocks

- building-block1: a original via
- building-block2: a redundant via
- building-block3: a original via and a redundant via
- building-block4: two original vias
- building-block5: two redundant vias
- building-block6: a original via and a redundant via (diagonal)
- building-block7: two original vias (diagonal)
- building-block8: two redundant vias (diagonal)
- building-block9: six original/redundant vias
Combinations of Building-Blocks

- Combinations of building-blocks to form guiding templates
Building-Blocks Detection & Conflict Graph

- Conflict graph $CG (V, E)$
  - vertex $v \in V$ denotes a building-block,
  - edge $e_{ij} \in E$ is an edge and $E = (E_C - E_T) \cup E_O$. $E_C$, $E_T$ and $E_O$ are the sets of conflict edges, template edges and overlap edges.
Conflict Edges

- The distance between two building-blocks are less than resolution limit space $d_s$.

![Diagram showing conflict edges, overlap edges, and template edges.](image)

- **Conflict edge**
- **Overlap edge**
- **Template edge**
Overlap Edges

- Two building-blocks are overlapped.
Template Edges

- If building-blocks $i$ and $j$ with $e_{ij} \in E_C$ can be assigned to a guiding template without any design error.

![Diagram showing template edges and their relationships](image)

- Conflict edge
- Overlap edge
- Template edge
Solution Flow

DSA Guiding Template

Routing Layout

Optical resolution limit spacing $d_s$

Preprocessing

Find All Redundant/Dummy Via Candidates

Detect Building Blocks

Construct Conflict Graph

Local Optimal Solver

Integer Linear Programming Formulation

Initial Solution Generation

Unconstrained Nonlinear Programming Solver

Redundant/Dummy Via Insertion with Template Assignment
Constraints

\[ x_i + x_j \leq 1, \quad \forall e_{ij} \in E; \]

\[ E = (E_C - E_T) \cup E_O \]

- Conflict edge
- Template edge
- Overlap edge
Conflict Structure Constraint

- Template constraint
  - If two building-blocks $i$ and $j$ are connected by a template edge, then they may be assigned to the same guiding template, but not necessarily.
  - If both of building-blocks $i$ and $l$ connect with $k$ by template edges, then $i$, $k$, $l$ may not be assigned to a same guiding template.

- Conflict structure (CS)
  - Three bblocks $i$, $k$ and $l$, in which $e_{ik}$ and $e_{kl}$ are template edges and there does not exist any edge between $i$ and $l$.

\[ x_i + x_k + x_l \leq 2, \quad \forall (i, k, l) \in CS; \]
Integer Linear Programming (ILP)

- Objectives:
  - Maximize the number of inserted redundant vias
  - Maximize the number of patterned vias by DSA
  - Let $N_v$ and $N_r$ are the numbers of included vias and redundant vias by building-block $i$, and

  $$w_i = N_v + \beta \cdot N_r,$$

- ILP Formulation

$$\max_{x} \sum_{i \in V} w_i x_i$$

subject to:

$$x_i + x_j \leq 1, \quad \forall e_{ij} \in E;$$

$$x_i + x_k + x_l \leq 2, \quad \forall (i, k, l) \in CS;$$

$$x_i \in \{0, 1\}, \quad \forall i \in V.$$
• **Claim 1.** The ILP is equivalent to the DRDV problem.

\[
\begin{align*}
\text{max}_x & \quad \sum_{i \in V} w_i x_i \\
\text{s.t.} & \quad x_i + x_j \leq 1, \quad \forall e_{ij} \in E; \\
& \quad x_i + x_k + x_l \leq 2, \quad \forall (i, k, l) \in CS; \\
& \quad x_i \in \{0, 1\}, \quad \forall i \in V.
\end{align*}
\] (1)

• Transfer inequality constraints to equality constraints.

\[
\begin{align*}
\text{max}_x & \quad \sum_{i \in V} w_i x_i \\
\text{s.t.} & \quad x_i x_j = 0, \quad \forall e_{ij} \in E; \\
& \quad x_i x_k x_l = 0, \quad \forall (i, k, l) \in CS; \\
& \quad x_i \in \{0, 1\}, \quad \forall i \in V.
\end{align*}
\] (2)
Equality Constraints

- Relax equality constraints to objective function.

\[
\begin{align*}
\max_\mathbf{x} & \quad \sum_{i \in V} w_i x_i \\
\text{s.t.} & \quad x_i x_j = 0, \quad \forall e_{ij} \in E; \\
& \quad x_i x_k x_l = 0, \quad \forall (i, k, l) \in CS; \\
& \quad x_i \in \{0, 1\}, \quad \forall i \in V,
\end{align*}
\]
Adjacent Matrix & CS Tensor

- Handle adjacent matrix and CS tensor.

\[
\max_x \sum_{i \in V} \{w_i x_i \prod_{j \in V} (1 - x_j) \prod_{k, l \in V} (1 - x_k x_l)\}
\]

\[
\text{s.t. } x_i \in \{0, 1\}, \forall i \in V.
\]

\[
\max_x \sum_{i \in V} \{w_i x_i \prod_{j \in V} (1 - x_j) \prod_{k, l \in V} (1 - x_k x_l) C_{ijkl}\}
\]

\[
\text{s.t. } x_i \in \{0, 1\}, \forall i \in V.
\]

\[
B_{ij} = \begin{cases} 1, & e_{ij} \in E \\ 0, & e_{ij} \notin E \end{cases}
\]

\[
C_{ijkl} = \begin{cases} 1, & (i, k, l) \in CS \\ 0, & (i, k, l) \notin CS \end{cases}
\]
Unconstrained Nonlinear Programming (UNP)

\[
\begin{align*}
\max_x & \quad \sum_{i \in V} \{ w_i x_i \prod_{j \in V} (1 - x_j)^{B_{ij}} \prod_{k, l \in V} (1 - x_k x_l)^{C_{ikl}} \} \\
\text{s.t.} & \quad x_i \in \{0, 1\}, \forall i \in V.
\end{align*}
\]

\[
\begin{align*}
x_i & \approx \sigma(y_i) = (1 + e^{-\gamma y_i})^{-1} \\
x_i & = \begin{cases} 
1, & y_i \geq 0 \\
0, & y_i < 0
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\max_y & \quad f(y) = \sum_{i \in V} \{ w_i \sigma(y_i) \prod_{j \in V} (1 - \sigma(y_j))^{B_{ij}} \prod_{k, l \in V} (1 - \sigma(y_k) \sigma(y_l))^{C_{ikl}} \}
\end{align*}
\]
\[
\max_y f(y) = \sum_{i \in V} \left\{ w_i \sigma(y_i) \prod_{j \in V} (1 - \sigma(y_j))^{B_{ij}} \prod_{k, l \in V} (1 - \sigma(y_k)\sigma(y_l))^{C_{ikl}} \right\}
\]

**Input:** A connected component of \( CG(V, E, W) \), convergence threshold \( \delta = 10^{-4} \);

**Output:** Solution \( x^* \) of ILP (2);

1. Initialize \( t \leftarrow 0 \);
2. Generate \( x^{(0)} \);
3. If \( x_i^{(0)} = 1 \), let \( y_i^{(0)} \leftarrow 1 \); otherwise, let \( y_i^{(0)} \leftarrow -1 \);
4. \textbf{repeat}
5. \quad \forall i \in V, \text{ compute } g_i^{(t)} ;
6. \quad \text{ Obtain } \nabla f(y^{(t)}) ;
7. \quad \alpha \leftarrow \text{LineSearch}(y^{(t)}) ;
8. \quad y^{(t+1)} \leftarrow y^{(t)} + \alpha \nabla f(y^{(t)}) ;
9. \quad t \leftarrow t + 1 ;
10. \quad \textbf{until } ||\nabla f(y^{(t)})|| < \delta
11. Get \( x_i^* \) by rounding \( \sigma(y_i^{(t)}) \) to the nearest integer, \( \forall i \in V \).

\[ \Theta(\max\{|V| \cdot |E|, |V| \cdot \|C\|_0|) \]
Local Optimal Convergence

\[ g_i(y) = \sigma(y_i) \prod_{j \in V} (1 - \sigma(y_j))^{B_{ij}} \prod_{k,l \in V} (1 - \sigma(y_k)\sigma(y_l))^{C_{ikl}} \]

\[ [\nabla f(y^{(t)})]_i = \frac{\partial f(y^{(t)})}{\partial y_i} \]

\[ = y_{wi}g_i^{(t)} \{(1 - \sigma(y_i^{(t)})) - \sum_{j} B_{ij}\sigma(y_j^{(t)}) - \sum_{k} \sum_{l} C_{ikl} \frac{\sigma(y_k^{(t)})(1 - \sigma(y_k^{(t)}))\sigma(y_l^{(t)})}{1 - \sigma(y_k^{(t)})\sigma(y_l^{(t)})} \} \]

- **Lemma 1.** Under above Equations, \( \sum_{i} w_i \Delta g_i \geq 0 \).

- **Theorem 1.** Under above Equations, \( f(y) \) does not decrease.

- **Corollary 1.** Strict inequality \( \sum_{i} w_i \Delta g_i > 0 \) cannot be achieved.

- **Theorem 2.** Our UNP solver converges to a local maximum.
Outline

- Introductions
- Problem Formulation
- Algorithm
- Experimental Results
- Conclusions
Experimental Settings

- **Platform**
  - C++ programming language
  - Unix machine with Intel Core 2.70 GHz CPU and 8 GB memory
  - ILP solver: CPLEX

- **Benchmarks**
  - 11 circuits are provided by Prof. Fang, modified from MCNC benchmarks and an industry Faraday benchmarks

- **Algorithms**
  - TCAD’17: DSA+RVI, ILP+Speed-up
  - ASPDAC’17: DSA+RVI+DVI, ILP+Speed-up
  - TVLSI’18: DSA+RVI+DVI, Two Stage MWIS Solver
  - Ours: DSA+RVI+DVI, UNP Solver

- **Indicators**
  - Manufacture rate, insertion rate, runtime
The Number of Vias

- The numbers of vias of benchmarks range from eight thousand to seventy thousand.

![The Number of Vias](image-url)
Comparison: Manufacture Rate

- Compared with “TCAD’17,” “ASPDAC’17,” and “TVLSI’18,” our algorithm achieves 6%, 0%, and 3% improvement on manufacture rate.


Comparison: Insertion Rate

- Compared with “TCAD’17,” “ASPDAC’17,” and “TVLSI’18,” our algorithm achieves 7%, 0%, and 2% improvement on insertion rate.
Comparison: Runtime

- Our algorithm is **3.99X and 13.32X** faster than “TCAD’17”, “ASPDAC’17”.

![Comparison chart showing CPU(s) for different benchmarks and algorithms.](chart.png)
Conclusions

- We introduce a building-block based manner instead of guiding template candidate to express solution.
- We proposed a general ILP formulation and relaxed it to an UNP. Furthermore, we develop a first-order optimization method to solve the UNP, which is a local optimal algorithm.
- Experimental results verify our algorithm achieves comparable experimental results with a state-of-the-art work, and saves 92% runtime.
Handle Guiding Template Cost

- A building-block would be assigned to a guiding template.
- A guiding template composed of one or two building-blocks.
- Assign proper weights to building-block $w_i$ ($\forall i \in V$) and corresponding template edge $\tilde{w}_{ij}$ ($\forall e_{ij} \in E_T$).

$$\max_x \sum_{i \in V} w_i x_i + \frac{\lambda}{2} \sum_{e_{ij} \in E_T} \tilde{w}_{ij} x_i x_j$$

s.t.
- $x_i + x_j \leq 1$, $\forall e_{ij} \in E$;
- $x_i + x_k + x_l \leq 2$, $\forall (i, k, l) \in CS$;
- $x_i \in \{0, 1\}$, $\forall i \in V$. 

(1')
Handle Guiding Template Cost

\[
\begin{align*}
\max_x & \sum_{i \in V} w_i x_i \{1 + \frac{\lambda}{2w_i} \sum_{j \in V, e_{ij} \in E_T} \tilde{w}_{ij} x_j\} \\
\text{s.t.} & \quad x_i x_j = 0, \quad \forall e_{ij} \in E; \\
& \quad x_i x_k x_l = 0, \quad \forall (i, k, l) \in CS; \\
& \quad x_i \in \{0, 1\}, \quad \forall i \in V,
\end{align*}
\]  

\[(2')\]

\[
\begin{align*}
\max_x & \sum_{i \in V} w_i x_i \{1 + \frac{\lambda}{2w_i} \sum_{j \in V, e_{ij} \in E_T} \tilde{w}_{ij} x_j\} \prod_{j \in V} (1 - x_j) \prod_{k, l \in V} (1 - x_k x_l) \\
\text{s.t.} & \quad x_i \in \{0, 1\}, \quad \forall i \in V.
\end{align*}
\]  

\[(3')\]
Thank You!