Layout Decomposition for Triple Patterning Lithography

Bei Yu[1] Kun Yuan[2] Boyang Zhang[1] Duo Ding[1]
David Z. Pan[1]

[1]ECE Dept. University of Texas at Austin, Austin, TX USA

[2]Cadence Design Systems, Inc., San Jose, CA USA

Nov. 7, 2011

Outline

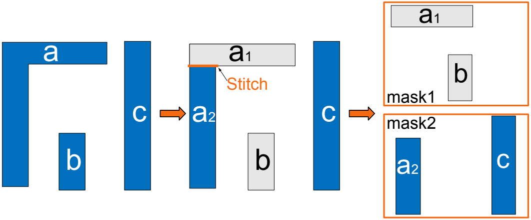
1. Introduction
2. Algorithm
   - TPL Decomposition Flow
   - Mathematical Formulation and Graph Simplification
   - Semidefinite Programming (SDP) Approximation
3. Experimental Results

Overcome the lithography limitations

- 193*nm* based lithography tool, hard for sub-30*nm*
- Delay or limitations of other techniques, i.e. EUV, E-Beam

Double/Multiple Patterning Lithography

- Original layout is divided into two/several masks (layout decomposition)
- Decrease pattern density, improve the depth of focus (DOF)
- Objective: minimize both conflicts and stitches



Triple Patterning Lithography (TPL)

- Layout is decomposed into three masks
- Similar but more difficult than 3 coloring problem

Why Triple Patterning Lithography (TPL) ?

- Resolve some native conflict from DPL
- Reduce the number of stitches
- Triple effective pitch, achieve further feature-size scaling ($22nm/16nm$)

Layout Decomposition

DPL Layout Decomposition

- Iterative Method (remove conflict → minimize stitch) Local Optimal
  - Cut based methodologies (ICCAD'08, ICCAD'09, ASPDAC'2010)
- Minimize conflict and stitch simultaneously
  - ILP Formulation (Yuan et. al ISPD'2009) → optimal but slow
  - Heuristic (Xu et. al ISPD'2010) → only for planar layout

TPL Layout Decomposition

- Previously only via layout is considered (Cork et. al SPIE'08)

TPL Layout Decomposition

Our work is the first systematic study for general layout

- Mathematical Formulation
- Novel Color representations
- Semidefinite Programming based approximation

TPL Layout Decomposition is HARDER

- Solution space is much bigger
- Conflict graph is NOT planar
- Detect conflict is not P, but NP-Complete

Problem Formulation

Problem: TPL Layout Decomposition

Input: layout and minimum coloring space.
Output: decomposed layout,
minimize the stitch number and the conflict number.

Two Lemmas:

- Deciding whether a *planar* graph is 3-colorable is NP-complete
- Coloring a 3-colorable graph with 4 colors is NP-complete

Theorem 1

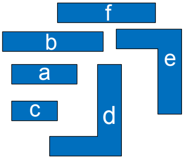
TPL Layout Decomposition problem is NP-Hard

Layout Graph and Decomposition Graph

Graphs Construction*:

1. Given input layout.
2. Generate Layout Graph (LG).
3. Projection.
4. Generate Decomposition Graph (DG).

* same with Yuan et. al ISPD'09



Input Layout

Two sets of edges:

- *CE*: conflict edge.
- *SE*: stitch edge.

Layout Graph and Decomposition Graph

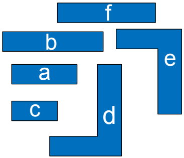
Graphs Construction*:

1. Given input layout.
2. Generate Layout Graph (LG).
3. Projection.
4. Generate Decomposition Graph (DG).

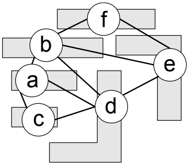
* same with Yuan et. al ISPD'09

Two sets of edges:

- *CE*: conflict edge.
- *SE*: stitch edge.



Input Layout



Layout Graph

Layout Graph and Decomposition Graph

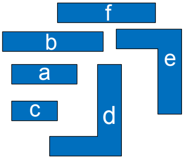
Graphs Construction*:

1. Given input layout.
2. Generate Layout Graph (LG).
3. Projection.
4. Generate Decomposition Graph (DG).

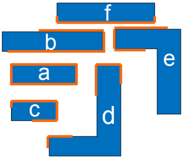
* same with Yuan et. al ISPD'09

Two sets of edges:

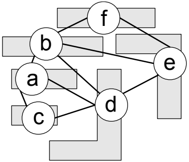
- *CE*: conflict edge.
- *SE*: stitch edge.



Input Layout



project



Layout Graph

Layout Graph and Decomposition Graph

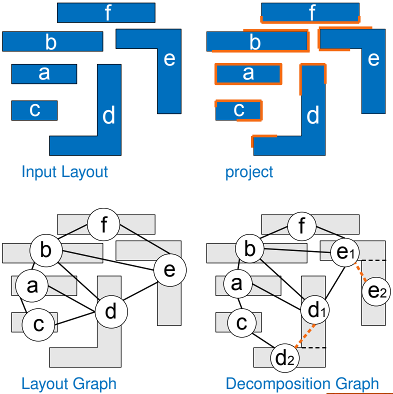
Graphs Construction*:

1. Given input layout.
2. Generate Layout Graph (LG).
3. Projection.
4. Generate Decomposition Graph (DG).

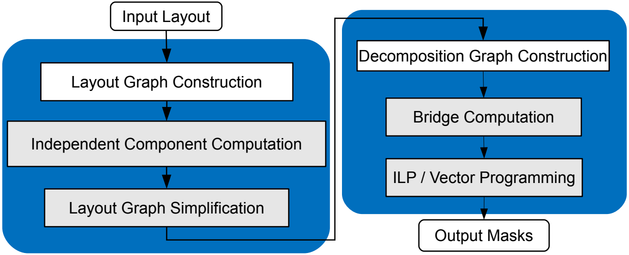
* same with Yuan et. al ISPD'09

Two sets of edges:

- *CE*: conflict edge.
- *SE*: stitch edge.

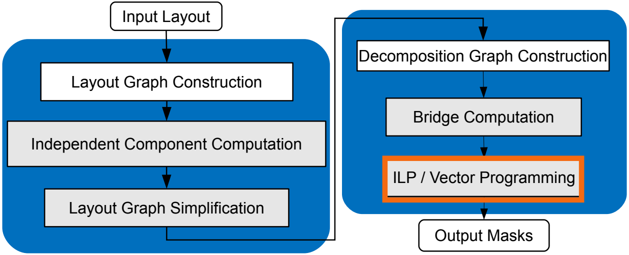


Overview of the TPL Decomposition Flow



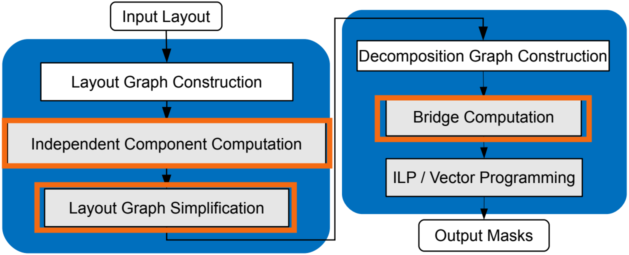
- Resolve Layout Decomposition problem:
  - Integer Linear Programming (ILP)
  - Vector Programming
- Three graph based Simplifications – improve scalability
- Vector Programming can be replaced by approximation methods:
  - Semidefinite Programming (SDP)
  - Mapping Algorithm

Overview of the TPL Decomposition Flow



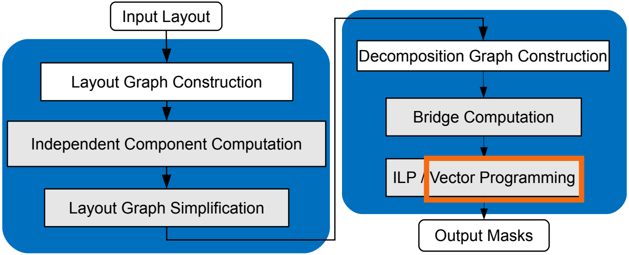
- Resolve Layout Decomposition problem:
  - Integer Linear Programming (ILP)
  - Vector Programming
- Three graph based Simplifications – improve scalability
- Vector Programming can be replaced by approximation methods:
  - Semidefinite Programming (SDP)
  - Mapping Algorithm

Overview of the TPL Decomposition Flow



- Resolve Layout Decomposition problem:
  - Integer Linear Programming (ILP)
  - Vector Programming
- Three graph based Simplifications – improve scalability
- Vector Programming can be replaced by approximation methods:
  - Semidefinite Programming (SDP)
  - Mapping Algorithm

Overview of the TPL Decomposition Flow



- Resolve Layout Decomposition problem:
  - Integer Linear Programming (ILP)
  - Vector Programming
- Three graph based Simplifications – improve scalability
- Vector Programming can be replaced by approximation methods:
  - Semidefinite Programming (SDP)
  - Mapping Algorithm

Mathematical Formulation

$$\begin{aligned}
\min \quad & \sum_{e_{ij} \in CE} c_{ij} + \alpha \sum_{e_{ij} \in SE} s_{ij} && && (1) \\
\text{s.t.} \quad & c_{ij} = (x_i == x_j) && \forall e_{ij} \in CE \\
& s_{ij} = x_i \oplus x_j && \forall e_{ij} \in SE \\
& x_i \in \{0, 1, 2\} && \forall i \in V
\end{aligned}$$

- $\sum c_{ij}$ is the number of conflicts, $\sum s_{ij}$ is the number of stitches
- Represent 3 colors using two 0-1 variables $(0,0), (0,1), (1,0)$
- Similar to previous DPL works, (1) can be transferred to ILP
- Solving ILP is NP-Hard problem, suffers from runtime penalty

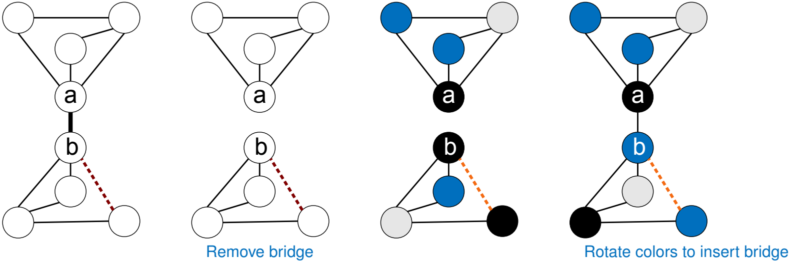
Graph Simplification

Independent Component Computation

- Partition the whole problem into several sub-problems

Bridge Computation

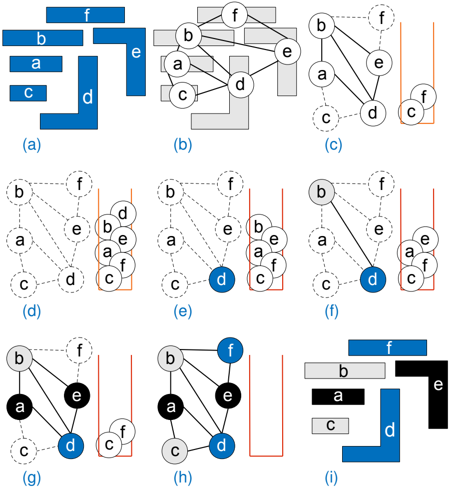
- Further partition the problem by removing bridges



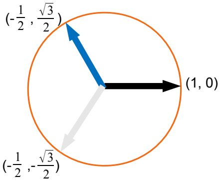
Graph Simplification (cont.)

Layout Graph Simplification

- Iteratively remove node with degree ≤ 2
- Push the nodes into stack
- Right layout can be directly colored



Vector Programming



***New* representation of colors**

- Three vectors $(1, 0), (-\frac{1}{2}, \frac{\sqrt{3}}{2})$ and $(-\frac{1}{2}, -\frac{\sqrt{3}}{2})$
- same color: $\vec{v_i} \cdot \vec{v_j} = 1$
- different color: $\vec{v_i} \cdot \vec{v_j} = -1/2$

Vector Programming:

$$\min \sum_{e_{ij} \in CE} \frac{2}{3}(\vec{v_i} \cdot \vec{v_j} + \frac{1}{2}) + \frac{2\alpha}{3} \sum_{e_{ij} \in SE} (1 - \vec{v_i} \cdot \vec{v_j}) \tag{2}$$

$$\text{s.t.} \quad \vec{v_i} \in \{(1, 0), (-\frac{1}{2}, \frac{\sqrt{3}}{2}), (-\frac{1}{2}, -\frac{\sqrt{3}}{2})\}$$

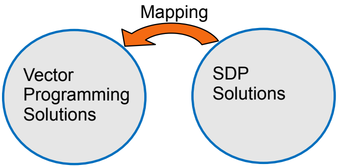
- Equal to Mathematical Formulation (1)
- Still NP-Hard

Semidefinite Programming (SDP) Approximation

Relax Vector Programming (2) to Semidefinite Programming (SDP)

$$\begin{aligned}\text{SDP:}\quad \min\quad & A \bullet X \\ & X_{ii} = 1, \quad \forall i \in V \\ & X_{ij} \geq -\frac{1}{2}, \quad \forall e_{ij} \in CE \\ & X \succeq 0\end{aligned} \tag{3}$$

SDP (3) can be solved in polynomial time



Mapping Algorithm

- Continuous SDP Solutions $\Rightarrow$ Three Vectors
- Tradeoff between speed and global optimality

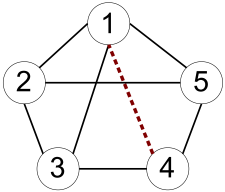
Example of SDP Approximation

$$A = \begin{pmatrix} 0 & 1 & 1 & -\alpha & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ -\alpha & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{aligned} \text{SDP:} \quad \min \quad & A \bullet X \qquad (3) \\ & X_{ii} = 1, \quad \forall i \in V \\ & X_{ij} \geq -\frac{1}{2}, \quad \forall e_{ij} \in CE \\ & X \succeq 0 \end{aligned}$$

After solving the SDP:

$$X = \begin{pmatrix} 1.0 & -0.5 & -0.5 & 1.0 & -0.5 \\ & 1.0 & -0.5 & -0.5 & -0.5 \\ & & 1.0 & -0.5 & 1.0 \\ & \dots & & 1.0 & -0.5 \\ & & & & 1.0 \end{pmatrix}$$



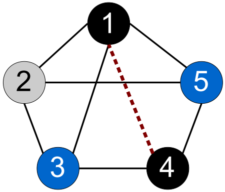
Example of SDP Approximation

$$A = \begin{pmatrix} 0 & 1 & 1 & -\alpha & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ -\alpha & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{aligned} \text{SDP:} \quad \min \quad & A \bullet X \\ & X_{ii} = 1, \quad \forall i \in V \\ & X_{ij} \geq -\frac{1}{2}, \quad \forall e_{ij} \in CE \\ & X \succeq 0 \end{aligned} \tag{3}$$

After solving the SDP:

$$X = \begin{pmatrix} 1.0 & -0.5 & -0.5 & 1.0 & -0.5 \\ & 1.0 & -0.5 & -0.5 & -0.5 \\ & & 1.0 & -0.5 & 1.0 \\ & \dots & & 1.0 & -0.5 \\ & & & & 1.0 \end{pmatrix}$$



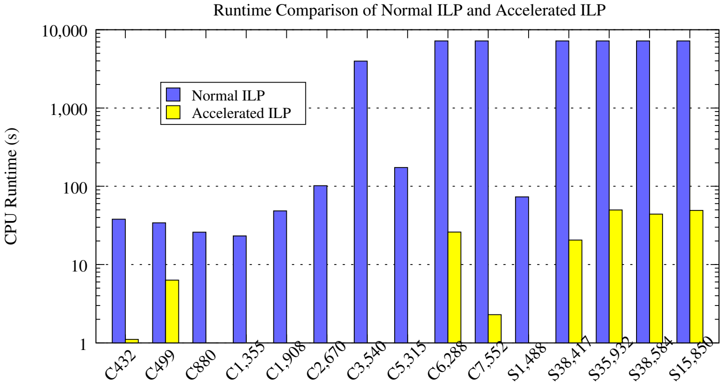
Experimental Results

Experimental Setting:

- implement in C++
- Intel Core 3.0GHz Linux machine with 32G RAM
- 15 layouts based on ISCAS-85 & 89 are tested

- Layout parser: OpenAccess2.2
- ILP solver: CBC
- SDP solver: CSDP

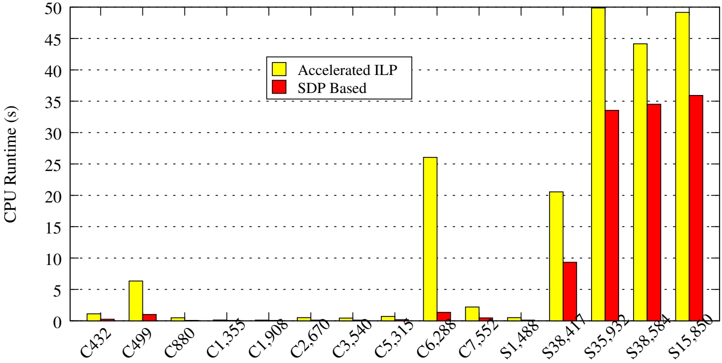
Experimental Results – Graph Simplification



- Graph Simplification can save 82% runtime [1]
- Still maintain the optimality

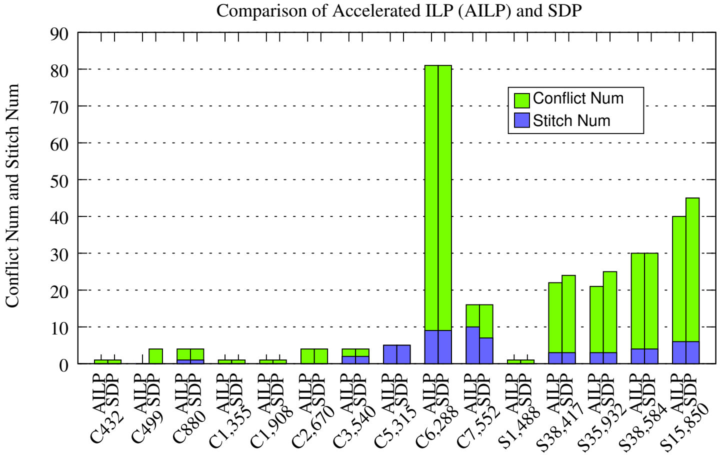
[1] Normal ILP uses Independent Component Computation

Experimental Results – How fast is SDP?



- SDP can effectively speed-up ILP
- Compared with Accelerated ILP, SDP can save 42% runtime

Experimental Results – How good (bad) is SDP?



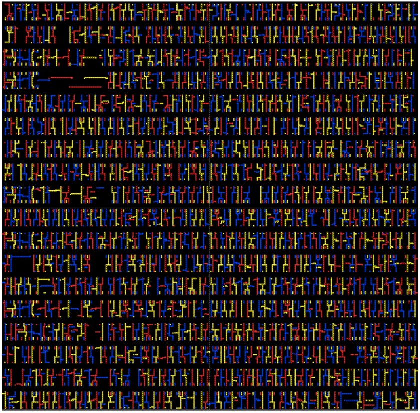
- SDP can achieve near optimal results.

Experimental Results – Dense Layout

Circuit	SE#	CE#	Accelerated ILP			SDP Based		
			st#	cn#	CPU(s)	st#	cn#	CPU(s)
C1	16	247	1	5	5.5	0	6	0.29
C2	38	289	0	15	17.32	0	16	0.77
C3	24	381	0	14	33.41	0	15	0.32
C4	56	437	9	32	203.17	9	32	0.49
avg.	-	-	2.5	16.5	64.9	2.25	17.3	0.468
ratio	-	-	1	1	1	0.9	1.05	0.007

- For very dense layout
- SDP can achieve 140× speed-up.

Experimental Results – S1488



- Stitch number: 0
- Conflict number: 1

Conclusion

- First systematic work on triple patterning layout decomposition
- Mathematical formulation to minimize both stitches and conflicts
- Novel color representations
- Semidefinite programming based approximation

Expect to see more researches on Triple Patterning Lithography

Thank You !

Appendix – ILP Formulation

$$\min \sum_{e_{ij} \in CE} c_{ij} + \alpha \sum_{e_{ij} \in SE} s_{ij} \tag{4}$$

$$
\begin{aligned}
\text{s.t. } & x_{i1} + x_{i2} \leq 1 & \\
& x_{i1} + x_{j1} \leq 1 + c_{ij1} & \forall e_{ij} \in CE \\
& (1 - x_{i1}) + (1 - x_{j1}) \leq 1 + c_{ij1} & \forall e_{ij} \in CE \\
& x_{i2} + x_{j2} \leq 1 + c_{ij2} & \forall e_{ij} \in CE \\
& (1 - x_{i2}) + (1 - x_{j2}) \leq 1 + c_{ij2} & \forall e_{ij} \in CE \\
& c_{ij1} + c_{ij2} \leq 1 + c_{ij} & \forall e_{ij} \in CE \\
& x_{i1} - x_{j1} \leq s_{ij1} & \forall e_{ij} \in SE \\
& x_{j1} - x_{i1} \leq s_{ij1} & \forall e_{ij} \in SE \\
& x_{i2} - x_{j2} \leq s_{ij2} & \forall e_{ij} \in SE \\
& x_{j2} - x_{i2} \leq s_{ij2} & \forall e_{ij} \in SE \\
& s_{ij} \geq s_{ij1}, s_{ij} \geq s_{ij2} & \forall e_{ij} \in SE
\end{aligned}
$$