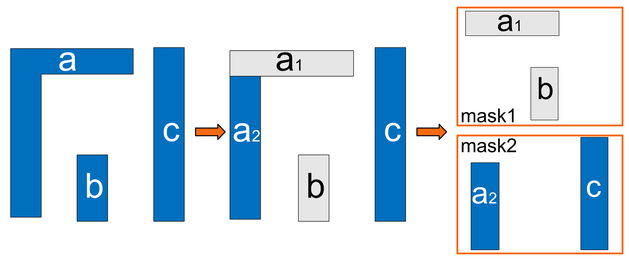
Layout Decomposition for Triple Patterning Lithography

Bei Yu, Kun Yuan*, Boyang Zhang, Duo Ding and David Z. Pan

ECE Dept. University of Texas at Austin, Austin, USA; *Cadence Design Systems, Inc., San Jose, USA

Introduction

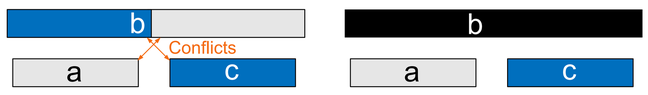


What's Layout Decomposition ?

- Original layout is divided into several masks
- Decrease pattern density in each exposure
- Improve resolution and depth of focus (DOF)
- Lots of works on Double Patterning Lithography (DPL)

Why Triple Patterning Lithography (TPL) ?

- Delay of next generation lithography (EUV)
- Original layout is divided into three masks (colors) to triple effective pitch
- Achieve further feature-size scaling (22*nm*/16*nm*)



Compared with Double Patterning Lithography (DPL), TPL can

- Resolve some native conflict from DPL
- Reduce the number of stitches

Previous Works

DPL Layout Decomposition

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

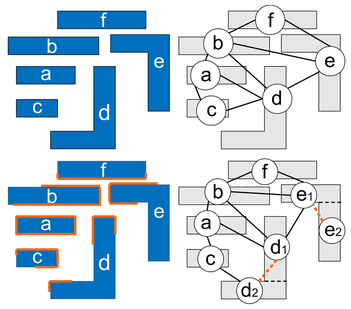
TPL Layout Decomposition is HARDER

- Conflict graph is NOT planar
- Detect conflict is not P, but NP-Complete
- Hard to use iterative strategy
- Solution space is much bigger

Problem Formulation

Graphs Construction:

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



Two sets of edges:

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

Problem: TPL Layout Decomposition

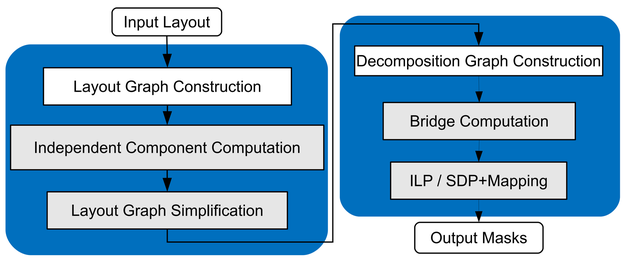
Given *DG*, *CE* and *SE*, assign all the nodes in the DG to three masks (colors) to minimize the stitch number and the conflict number

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**

Overview of the TPL Decomposition Flow



Three graph based Speed-Ups

Resolve 3-coloring problem:

- Integer Linear Programming (ILP)

ILP can be replaced by approximation methods:

- Semidefinite Programming (SDP)
- Mapping Algorithm

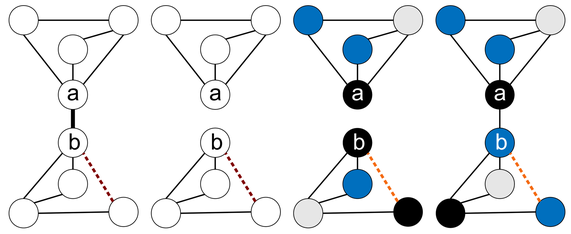
Speed-Up Techniques

Independent Component Computation

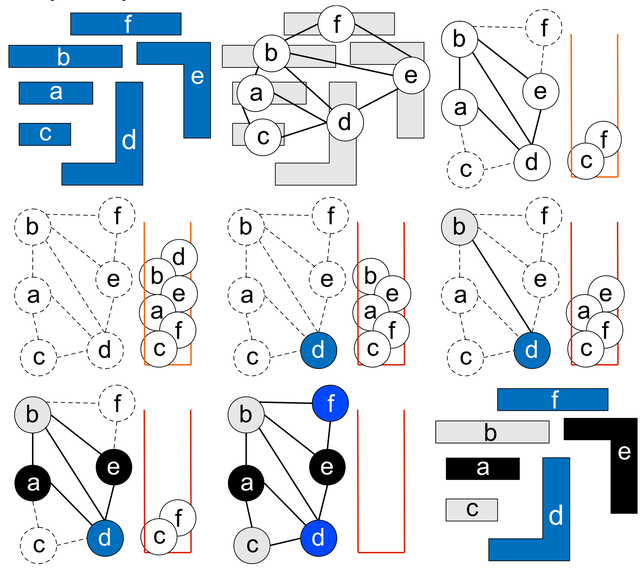
- Partition the whole problem into several sub-problems

Bridge Computation

- Further partition the problem by removing bridges



Layout Graph Simplification



Optimal Formulation (ILP)

Mathematical Formulation:

$$\min \sum_{e_{ij}\in CE} c_{ij} + \alpha \sum_{e_{ij}\in SE} s_{ij} \qquad (1)$$

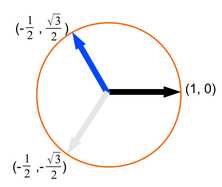
$$\text{s.t. } c_{ij} = (x_i == x_j) \quad \forall e_{ij} \in CE$$
$$s_{ij} = x_i \oplus x_j \quad \forall e_{ij} \in SE$$
$$x_i \in \{0,1,2\} \quad \forall i \in V$$

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

Semidefinite Programming (SDP) Approximation

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:

$$\min \sum_{e_{ij}\in CE} \frac{2}{3}(\vec{v}_i \cdot \vec{v}_j + \frac{1}{2}) + \frac{2\alpha}{3} \sum_{e_{ij}\in SE} (1 - \vec{v}_i \cdot \vec{v}_j) \qquad (2)$$

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

- Equal to Mathematical Formulation (1), still NP-Hard

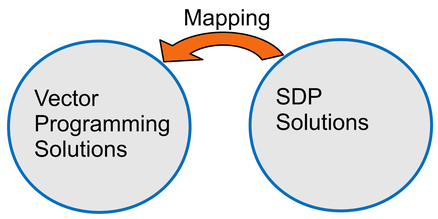
Relax Vector Programming (2) to Semidefinite Programming (SDP)

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

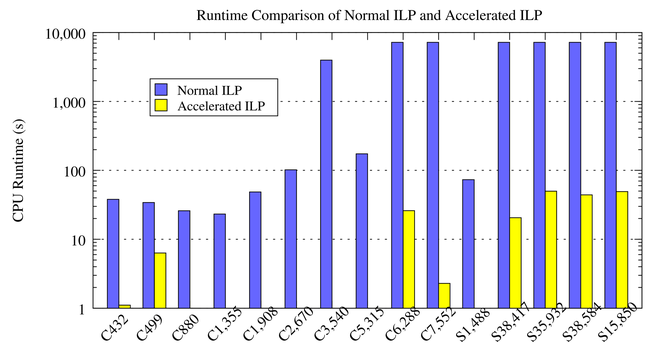
SDP (3) can be solved in polynomial time

Mapping Algorithm

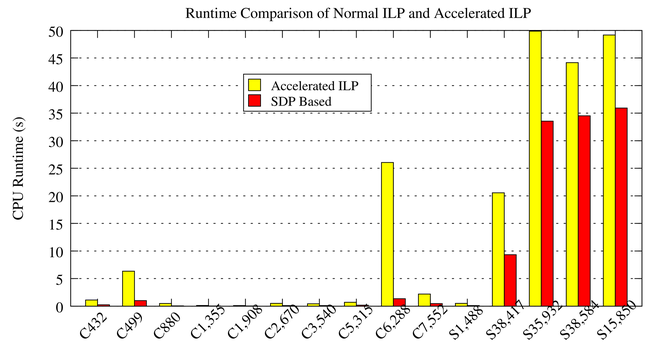
- Continuous SDP Solutions ⇒ Three Vectors
- Tradeoff between speed and global optimality



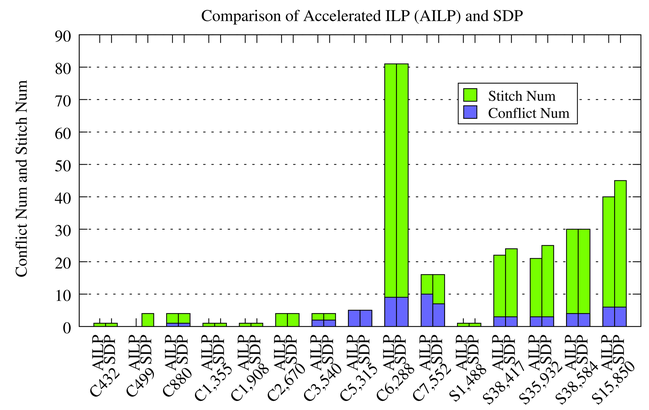
Experimental Results – ISCAS-85 & 89 layouts



- Normal ILP uses Independent Component Computation
- Graph Simplification is effective
- Still maintain the optimality



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



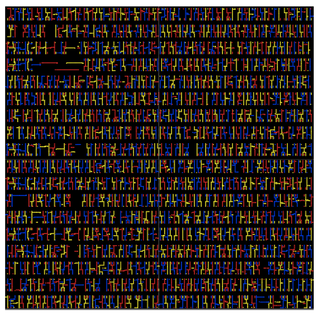
- SDP can achieve near optimal results

Experimental Results – very dense layouts

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

- SDP can achieve 140× speed-up

Decomposed Result



S1488 decomposed layout

- Stitch number: 0
- Conflict number: 1

Related work

Related work was accepted by ICCAD'2011 **(William J. McCalla Best Paper Award Finalist)**