Layout Decomposition for Quadruple Patterning Lithography and Beyond

Bei Yu, David Z. Pan

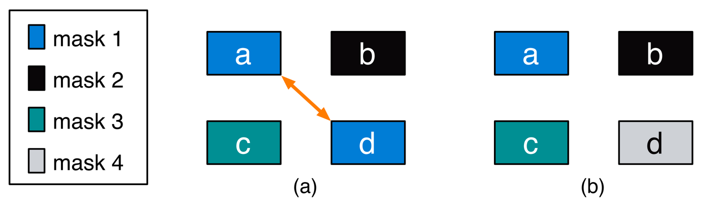
ECE Department, University of Texas at Austin, Austin, TX

Introduction

- Natural extension of triple patterning lithography (TPL)
- But with one more mask

Why QPL?

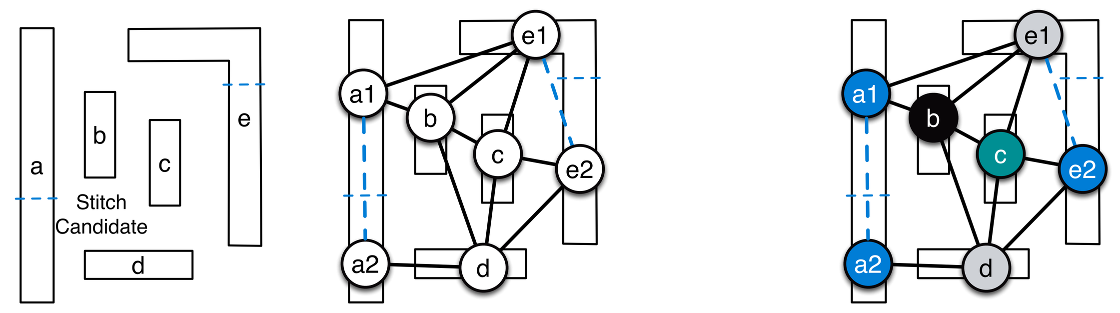
- Delay of EUVL
- Reserach perspective: need to be prepared
- Resolve native conflict from triple patterning



Problem Formulation

Input:

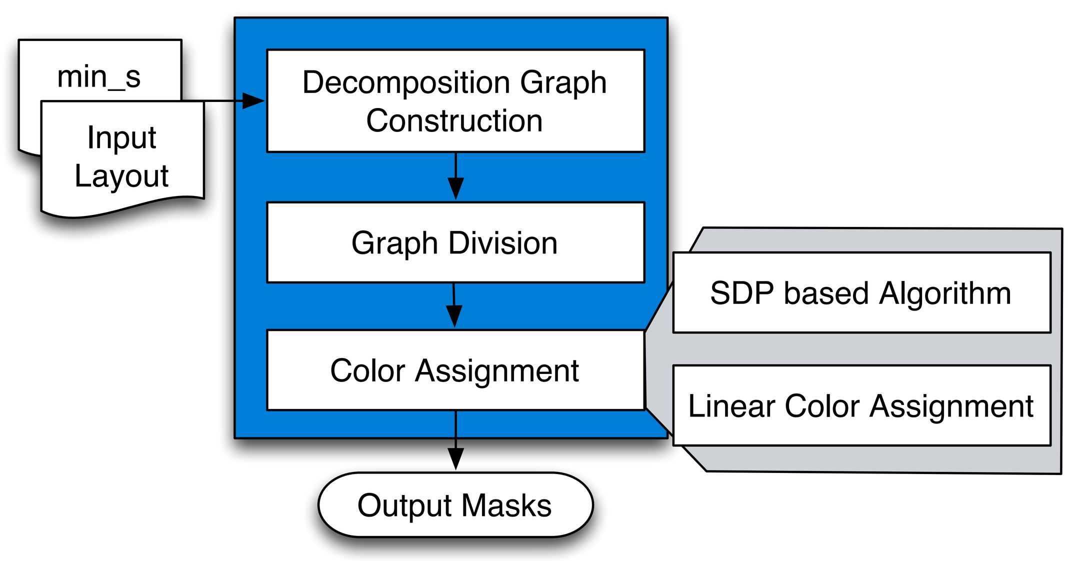
- Input layout patterns
- Minimum coloring distance min_s



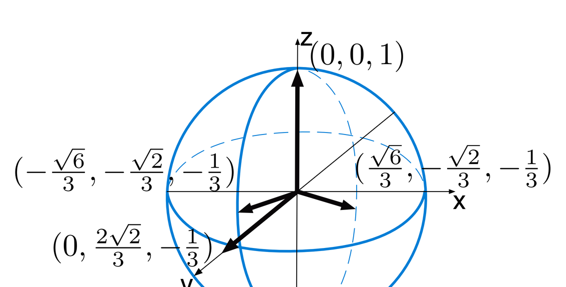
Output:

- Decomposed layout
- The conflict number & the stitch number

Overall Flow



SDP based Color Assignment



Vector based Color Representation

- Four vectors
- same color: $\vec{v}_i \cdot \vec{v}_j = 1$
- different color: $\vec{v}_i \cdot \vec{v}_j = -1/3$

Vector Programming:

$$\min \sum_{e_{ij}\in CE} \frac{3}{4}(\vec{v}_i \cdot \vec{v}_j + \frac{1}{3}) + \frac{3\alpha}{4} \cdot \sum_{e_{ij}\in SE} (1 - \vec{v}_i \cdot \vec{v}_j)$$
$$\text{s.t. } \vec{v}_i \in \{(0,0,1), (0, \frac{2\sqrt{2}}{3}, -\frac{1}{3}), (\frac{\sqrt{6}}{3}, -\frac{\sqrt{2}}{3}, -\frac{1}{3}), (-\frac{\sqrt{6}}{3}, -\frac{\sqrt{2}}{3}, -\frac{1}{3})\}$$

Relax to Semidefinite Programming (SDP)

$$\min \sum_{e_{ij}\in CE} \vec{v}_i \cdot \vec{v}_j - \alpha \sum_{e_{ij}\in SE} \vec{v}_i \cdot \vec{v}_j$$
$$\text{s.t. } \vec{v}_i \cdot \vec{v}_i = 1, \quad \forall i \in V$$
$$\vec{v}_i \cdot \vec{v}_j \geq -\frac{1}{3}, \quad \forall e_{ij} \in CE$$

Mapping: Continuous solutions to discrete $v_i \cdot v_j$

Algorithm: Backtrack based Mapping

Require: SDP solution x_{ij}, threshold value t_{th};

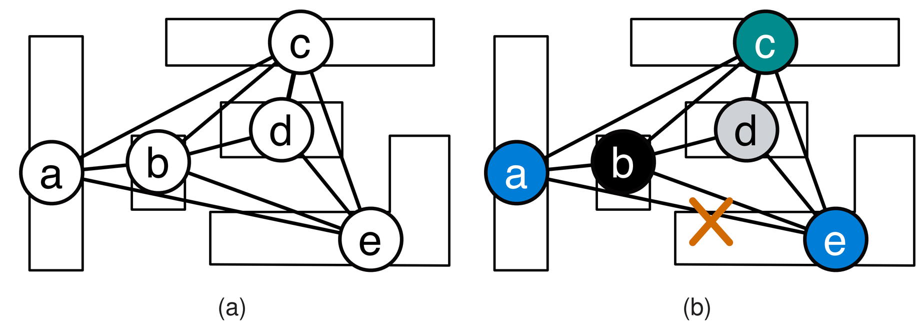
```
for all x_ij ≥ t_th do
    Combine vertices v_i, v_j into one larger vertex;
end for
Construct merged graph G' = {V', CE', SE'};
BACKTRACK(0, G');
return color assignment result in G';

function BACKTRACK(t, G')
  if t ≥ size[G'] then
      if Find a better color assignment then
          Store current color assignment;
      end if
  else
      for all legal color c do;
          G'[t] ← c;
          BACKTRACK(t + 1, G');
          G'[t] ← −1;
      end for
  end if
end function
```

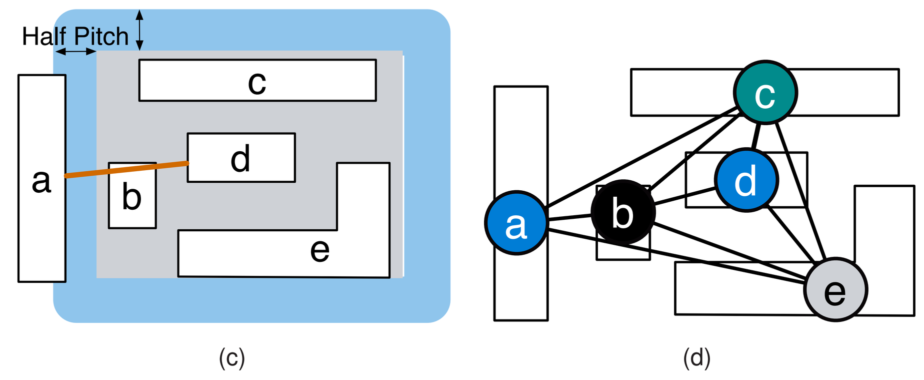
Linear Color Assignment

Technique One: Color-Friendly Rules

- Any coloring order (e.g. a-b-c-d-e) results in **Local Optimality**.

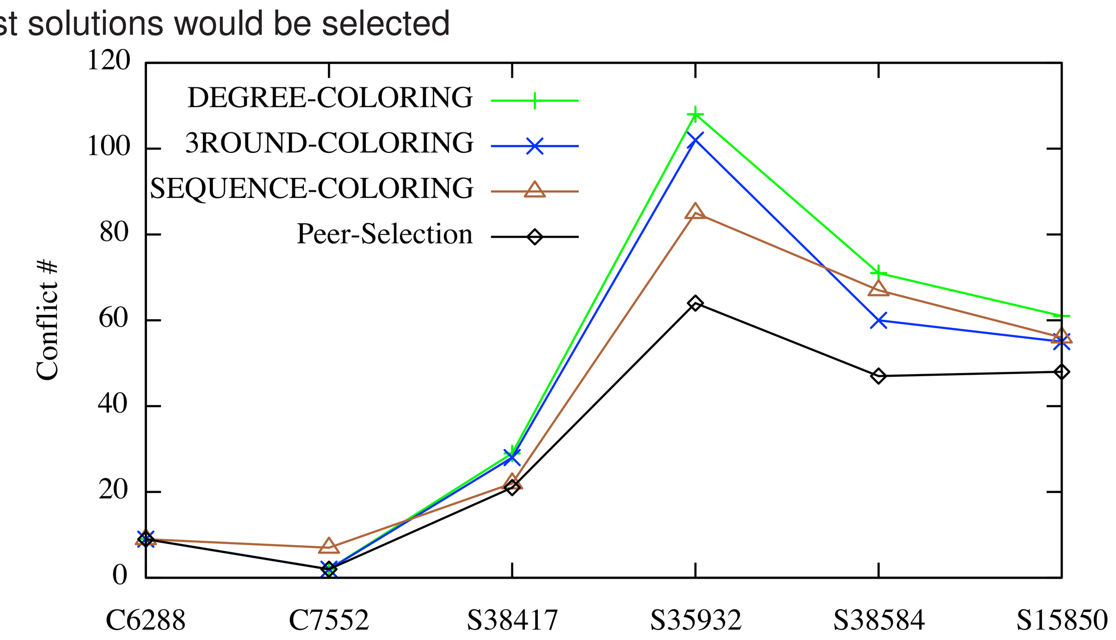


- a and d are **color-friendly**



Technique Two: Peer Selection

- Three orders would be processed simultaneously
- Best solutions would be selected



Algorithm: Linear Color Assignment

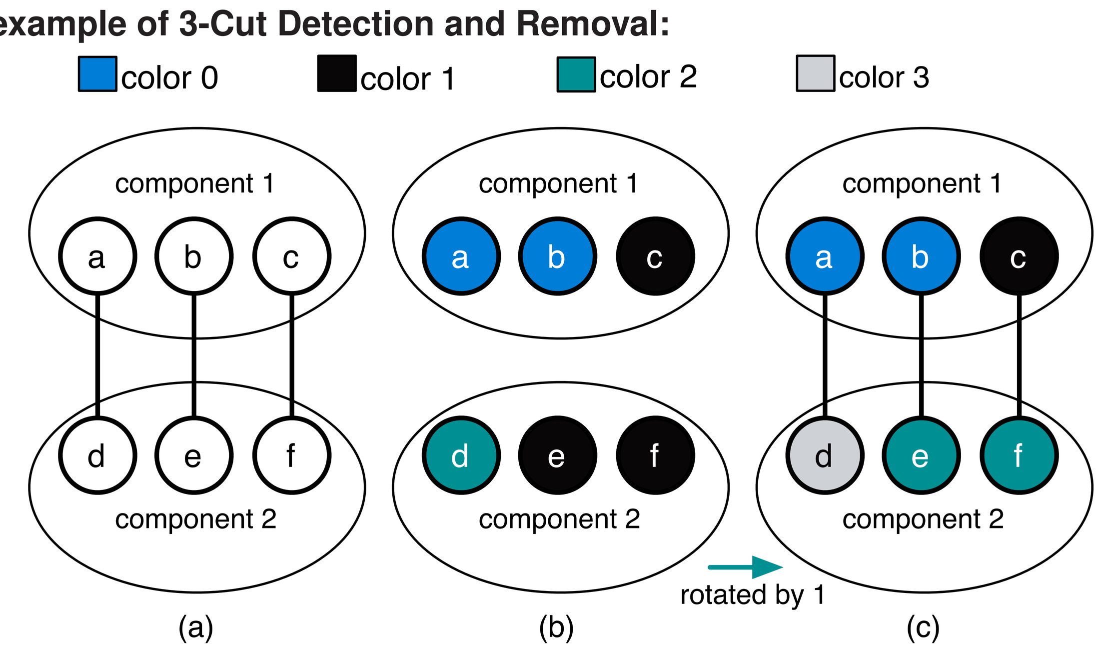
Require: Decomposition graph $G = \{V, CE, SE\}$, Stack S;

```
while ∃v_i ∈ V s.t. d_conf(v_i) < 4 & d_stit(v_i) < 2 do
    S.push(v_i);
    G.delete(v_i);
end while
Construct vector vec = {vec[1], vec[2], vec[3]};
C1 = SEQUENCE-COLORING(V);
C2 = DEGREE-COLORING(vec);
C3 = 3ROUND-COLORING(vec);
C = best coloring solution among {C1, C2, C3};
POST-REFINEMENT(V);
while !S.empty() do
    v_i = S.pop();
    G.add(v_i);
    c(v_i) ← a legal color;
end while
```

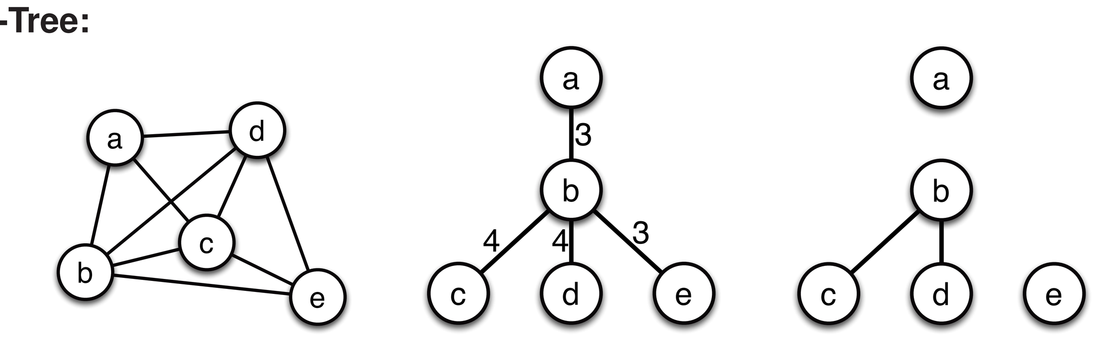
$O(n)$ runtime complexity!

GH-Tree based 3-Cut Removal

An example of 3-Cut Detection and Removal:



GH-Tree:



Algorithm: GH-tree based 3-Cut Removal

Require: Decomposition graph $G = \{V, CE, SE\}$;

```
Construct GH-tree;
Remove the edges with weight < 4;
Compute connected components on remaining GH-tree;
for each component do
    Color assignment on this component;
end for
Color rotation to interconnect all components;
```

General K-Patterning Layout Decomposition

Theorem

SDP formulation in (1) can provide $v_i \cdot v_j$ pairs for K-patterning color assignment problem.

$$\min \sum_{e_{ij}\in CE} (\vec{v}_i \cdot \vec{v}_j + \frac{1}{k-1}) + \alpha \sum_{e_{ij}\in SE} (1 - \vec{v}_i \cdot \vec{v}_j) \tag{1}$$
$$\text{s.t. } \vec{v}_i \cdot \vec{v}_i = 1, \quad \forall i \in V$$
$$\vec{v}_i \cdot \vec{v}_j \geq -\frac{1}{k-1}, \quad \forall e_{ij} \in CE$$

Theorem

For K-patterning layout decomposition problem, dividing graph through $(K-1)$-cut does not increase the final conflict number.

Experimental Results

- Using C++ on 3.0GHz Linux machine
- **CSDP** as SDP solver
- Benchmarks from triple patterning work [Yu+,ICCAD'11]

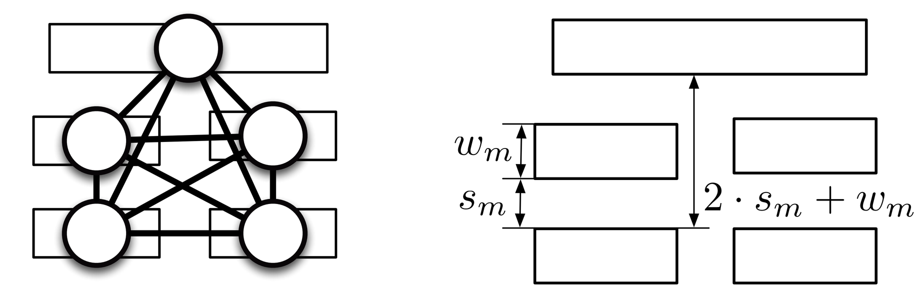
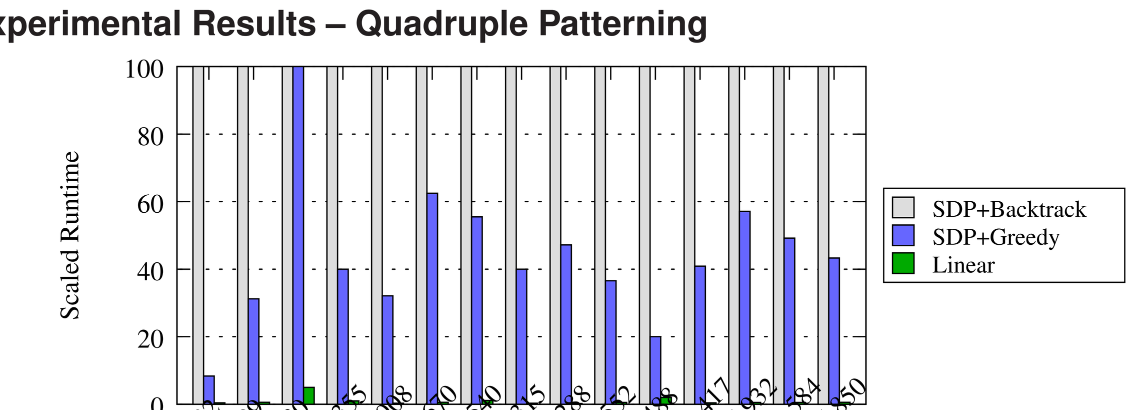


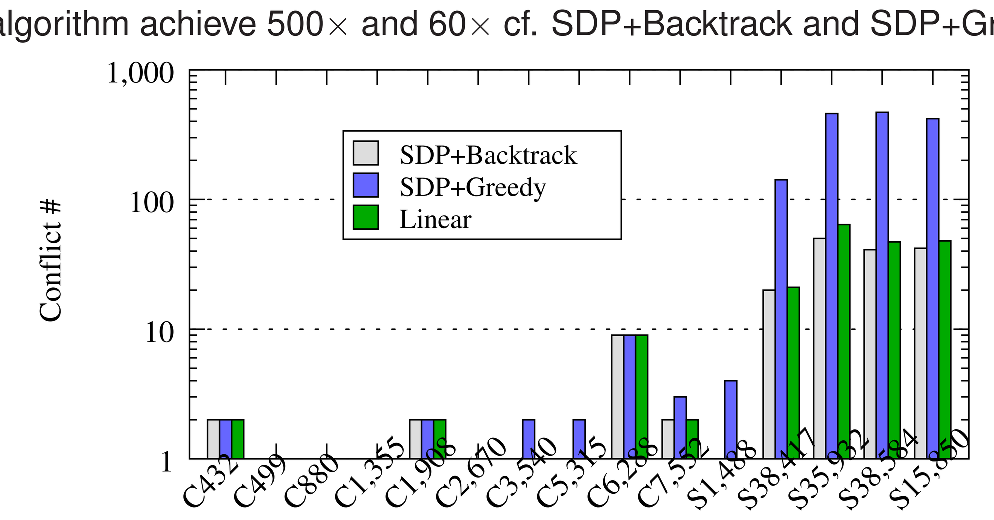
Figure: $min_s = 2 \cdot s_m + w_m$ may cause K_5 structure.

- For quadruple patterning: $min_s = 2 \cdot s_m + 2 \cdot w_m = 80$
- For pentuple patterning: $min_s = 3 \cdot s_m + 2.5 \cdot w_m = 110$

Experimental Results – Quadruple Patterning



- Linear algorithm achieve 500× and 60× cf. SDP+Backtrack and SDP+Greedy



- Linear algorithm gets similar conflict# cf. SDP+Backtrack

Experimental Results – Pentuple Patterning

Circuit	SDP+Backtrack			SDP+Greedy			Linear		
	cn#	st#	CPU(s)	cn#	st#	CPU(s)	cn#	st#	CPU(s)
C6288	19	2	2.4	19	2	0.49	19	5	0.005
C7552	1	1	0.3	1	1	0.05	1	4	0.001
S38417	0	4	1.45	0	4	0.21	0	4	0.001
S35932	5	20	8.11	5	20	0.62	5	25	0.009
S38584	3	4	1.66	7	3	0.3	3	6	0.008
S15850	6	5	2.7	7	5	0.4	5	15	0.007
avg.	5.7	6.0	2.77	6.5	5.83	0.35	5.5	9.8	0.005
ratio	**1.0**	**1.0**	**1.0**	**1.15**	**0.97**	**0.12**	**0.97**	**1.64**	**0.002**

Conclusions

- First layout decomposition framework for Quadrule Patterning and Beyond
- Our algorithm is effective and efficient to obtain high quality solution
- MPL may be a promising manufacturing solution for sub-10nm technology node
- Facilitating the advancement of MPL technology

Acknowledgements

- The authors would like to thank Tsung-Wei Huang for helpful discussions.
- This work is supported in part by Oracle, NSF, and NSFC.