Layout Decomposition for Quadruple Patterning Lithography and Beyond

Bei Yu, David Z. Pan

Department of Electrical & Computer Engineering
University of Texas at Austin, TX USA

06/03/2014

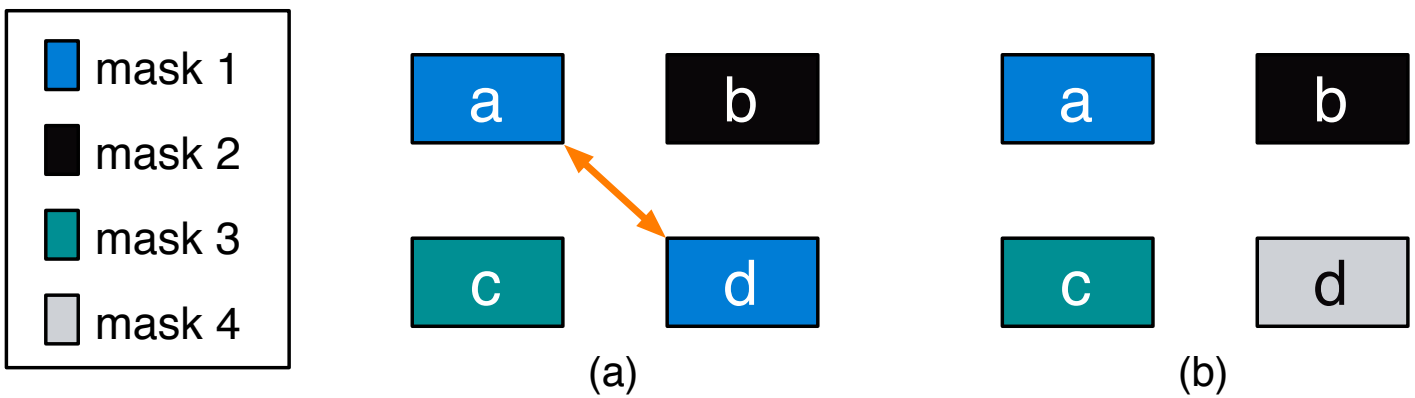
Supported by IBM scholarship, NSF, NSFC, SRC

Quadruple Patterning Lithography (QPL)

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

Why QPL?

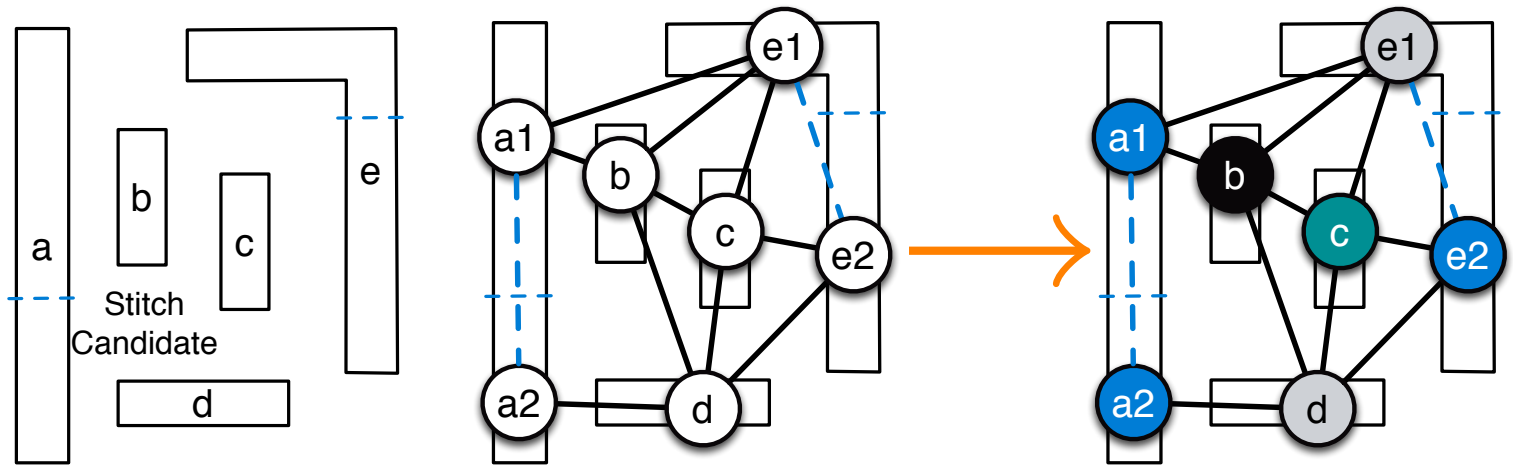
- Delay of EUVL
- CAD tools: need to be prepared
- Resolve native conflict from triple patterning



Problem Formulation

Input:

- Input layout patterns
- Minimum coloring distance min_s

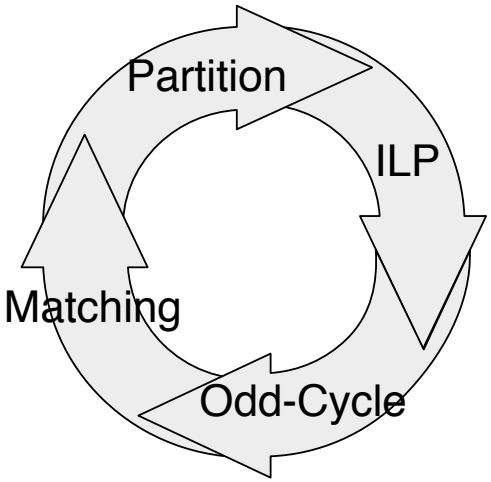


Output:

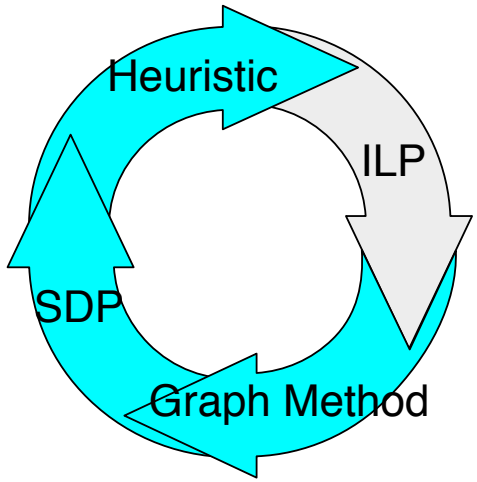
- Decomposed layout
- Minimize the conflict number & the stitch number

Layout Decomposition – From Double To Quadruple

Double Patterning

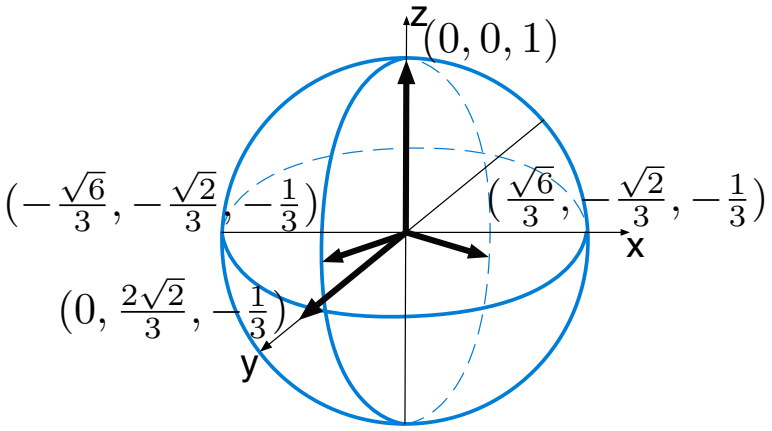


Triple Patterning



Quadruple Patterning ...

SDP Formulation with NEW Color Representations



- Four unit vectors
- same color: $\vec{v_i} \cdot \vec{v_j} = 1$
- different color: $\vec{v_i} \cdot \vec{v_j} = -1/3$

Semidefinite Programming (SDP) Formulation

$$\min \sum_{e_{ij} \in CE} \vec{v_i} \cdot \vec{v_j} - \alpha \sum_{e_{ij} \in SE} \vec{v_i} \cdot \vec{v_j}$$

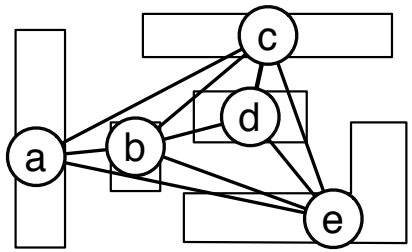
$$\text{s.t.} \quad \vec{v_i} \cdot \vec{v_i} = 1, \quad \forall i \in V$$

$$\vec{v_i} \cdot \vec{v_j} \geq -\frac{1}{3}, \quad \forall e_{ij} \in CE$$

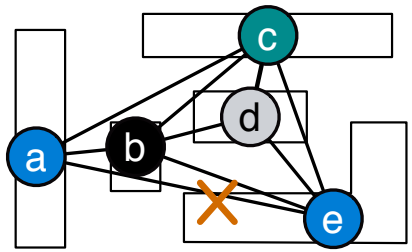
- Greedy Mapping v.s. Backtrack Mapping

New Linear Color Assignment

- Linear runtime complexity: resolved one by one
- But, Any coloring order results in **Local Optimality**
- Example: order a-b-c-d-e

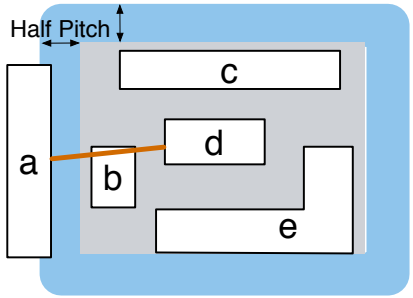


(a)

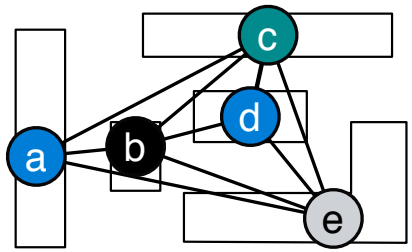


(b)

Color-Friendly Rules:



(c)

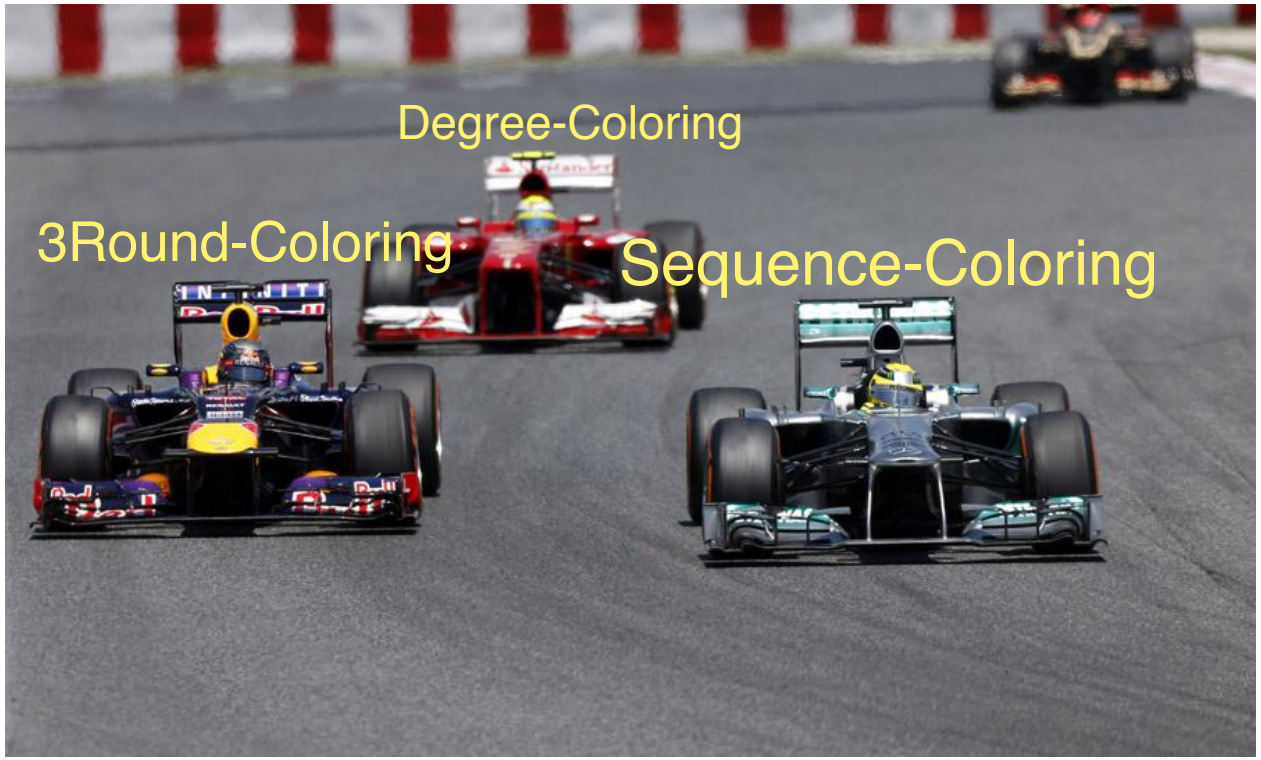


(d)

New Linear Color Assignment

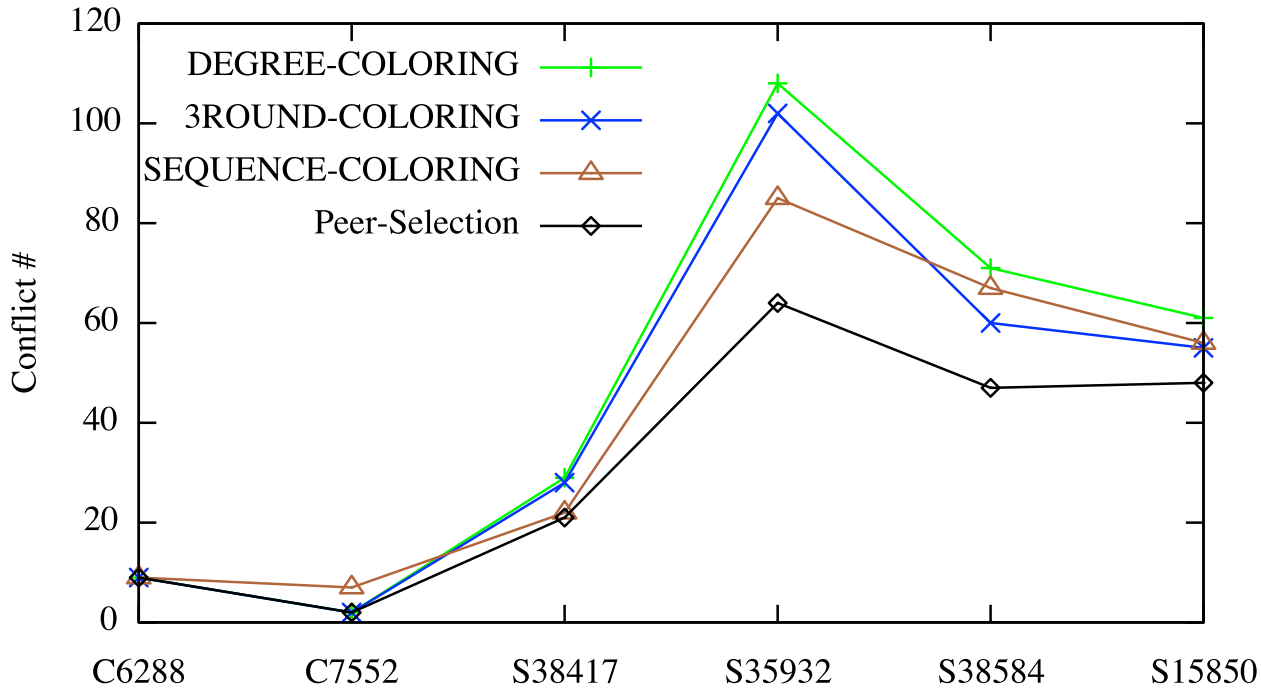
Peer Selection:

- Three orders would be processed simultaneously
- Best solution would be selected
- Still Linear runtime complexity



New Linear Color Assignment

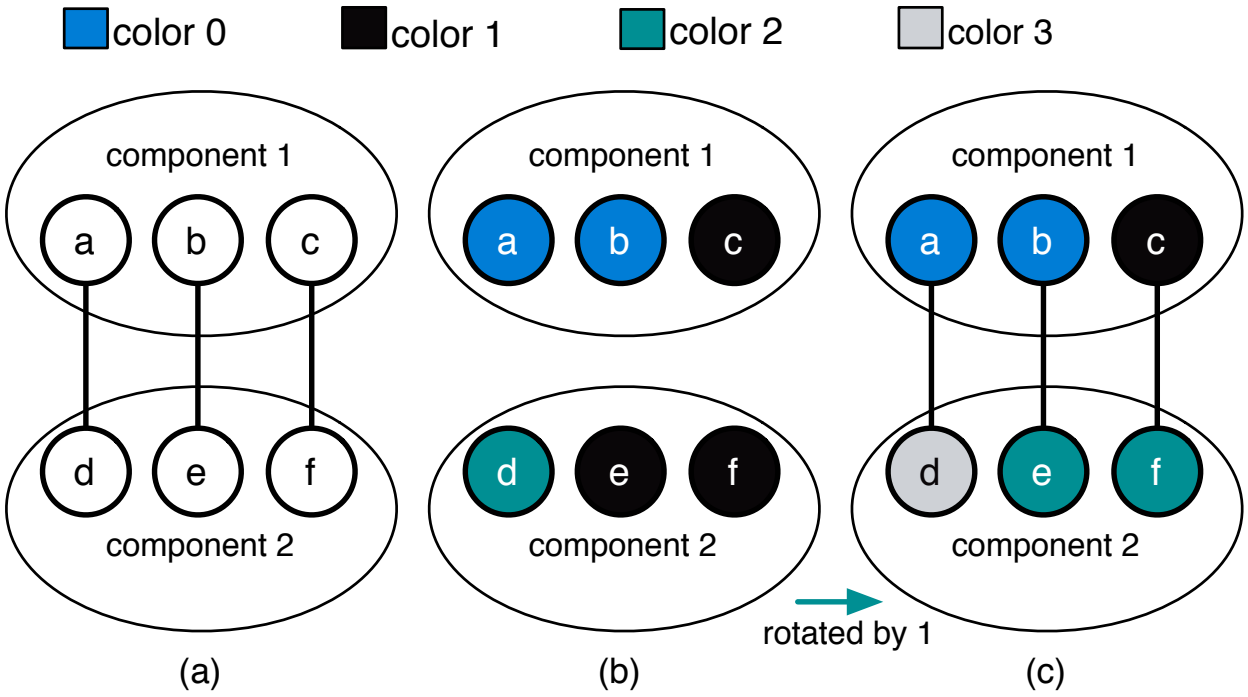
Peer Selection:



Better results than any single order

New: 3-Cut Removal

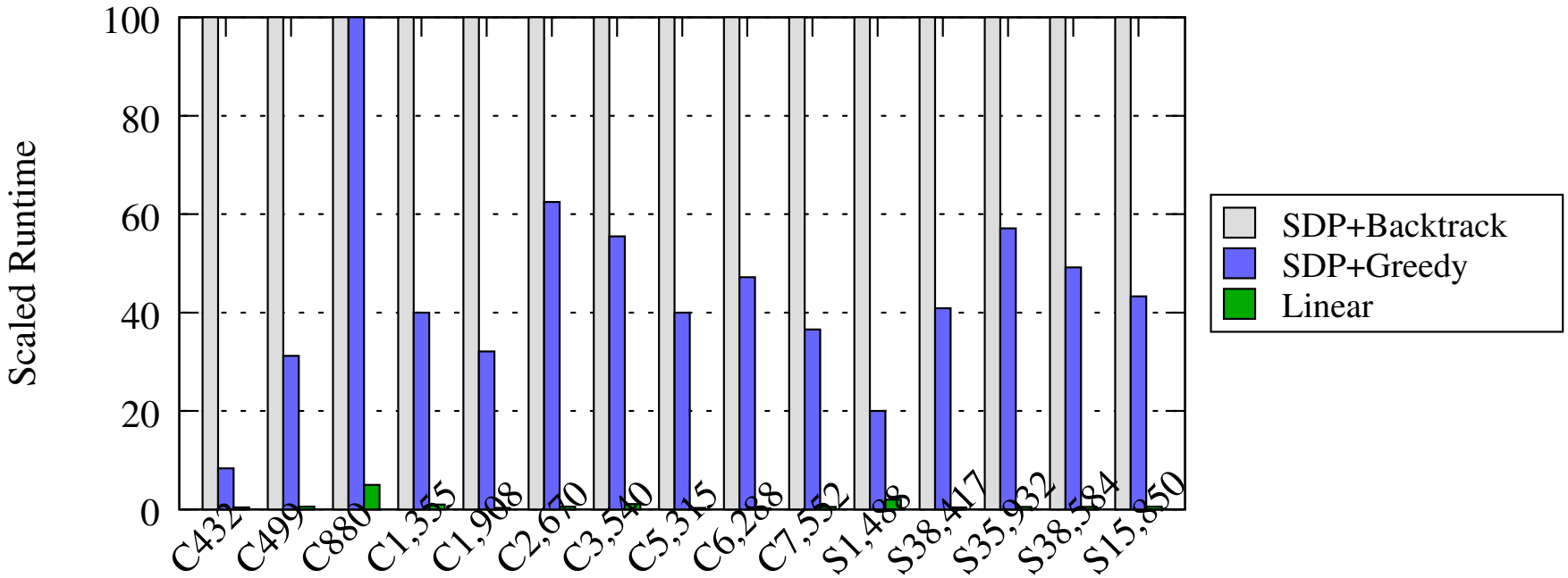
- Reduce the problem size
- Example:



GH-Tree to find 3-Cuts

Experimental Results – Quadruple Patterning

Runtime Comparison:

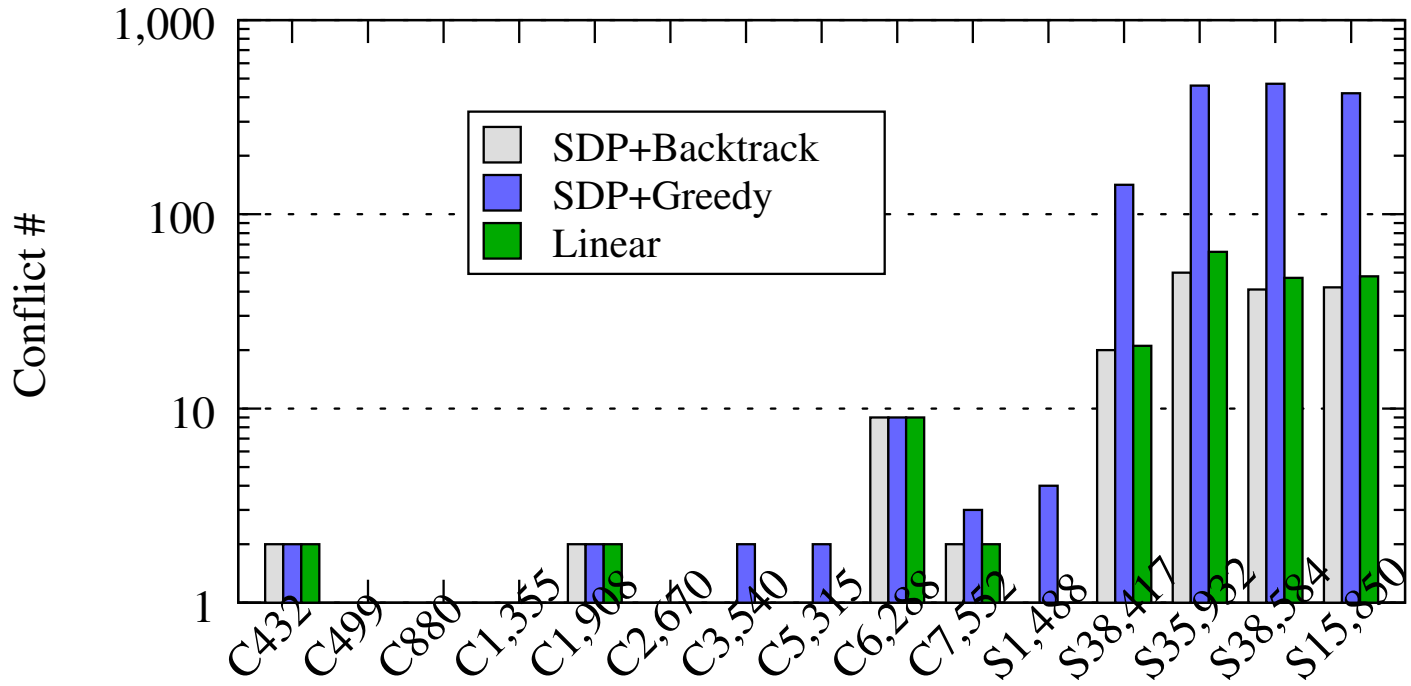


Linear algorithm achieves

- 500× cf. SDP+Backtrack
- 60× cf. SDP+Greedy

Experimental Results – Quadruple Patterning

Conflict # Comparison:



Linear algorithm achieves:

- Similar conflict # cf. SDP+Backtrack
- 67% Conflict # reduction cf. SDP+Greedy

Extend Algorithms to General K-Patterning

- Semidefinite programming (SDP)
- Linear color assignment
- GH-Tree based graph division

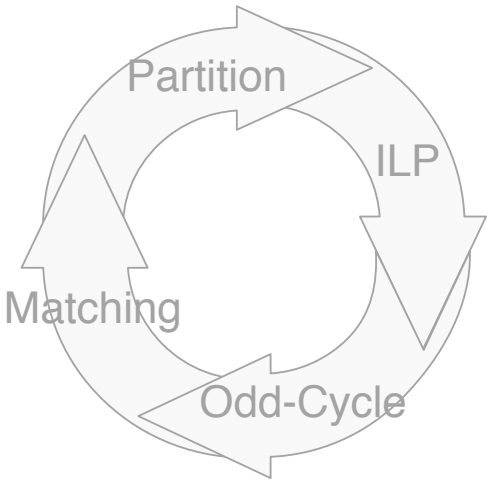
Results for Pentuple Patterning ($K = 5$)

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**

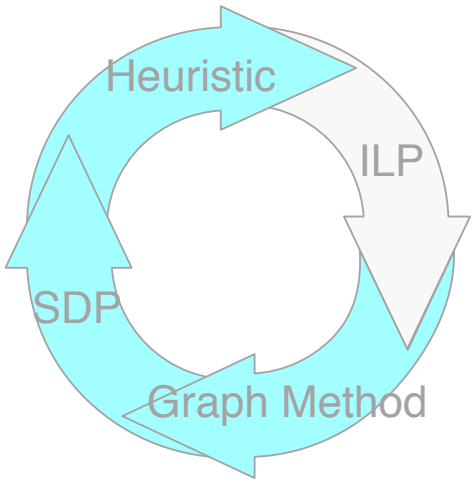
- Linear color assignment: best conflict #, 500$\times$ speed-up

Conclusions

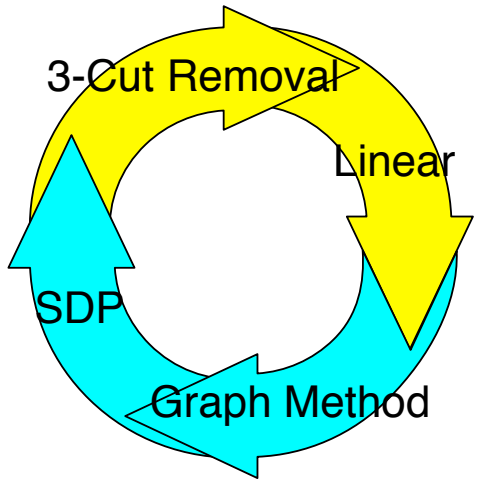
Double Patterning



Triple Patterning



Quadruple Patterning ...



- First attempt for Quadrule Patterning and Beyond
- Generic & Robust to General Patterning
- Facilitaing the advancement of Multiple Patterning

Thank You !