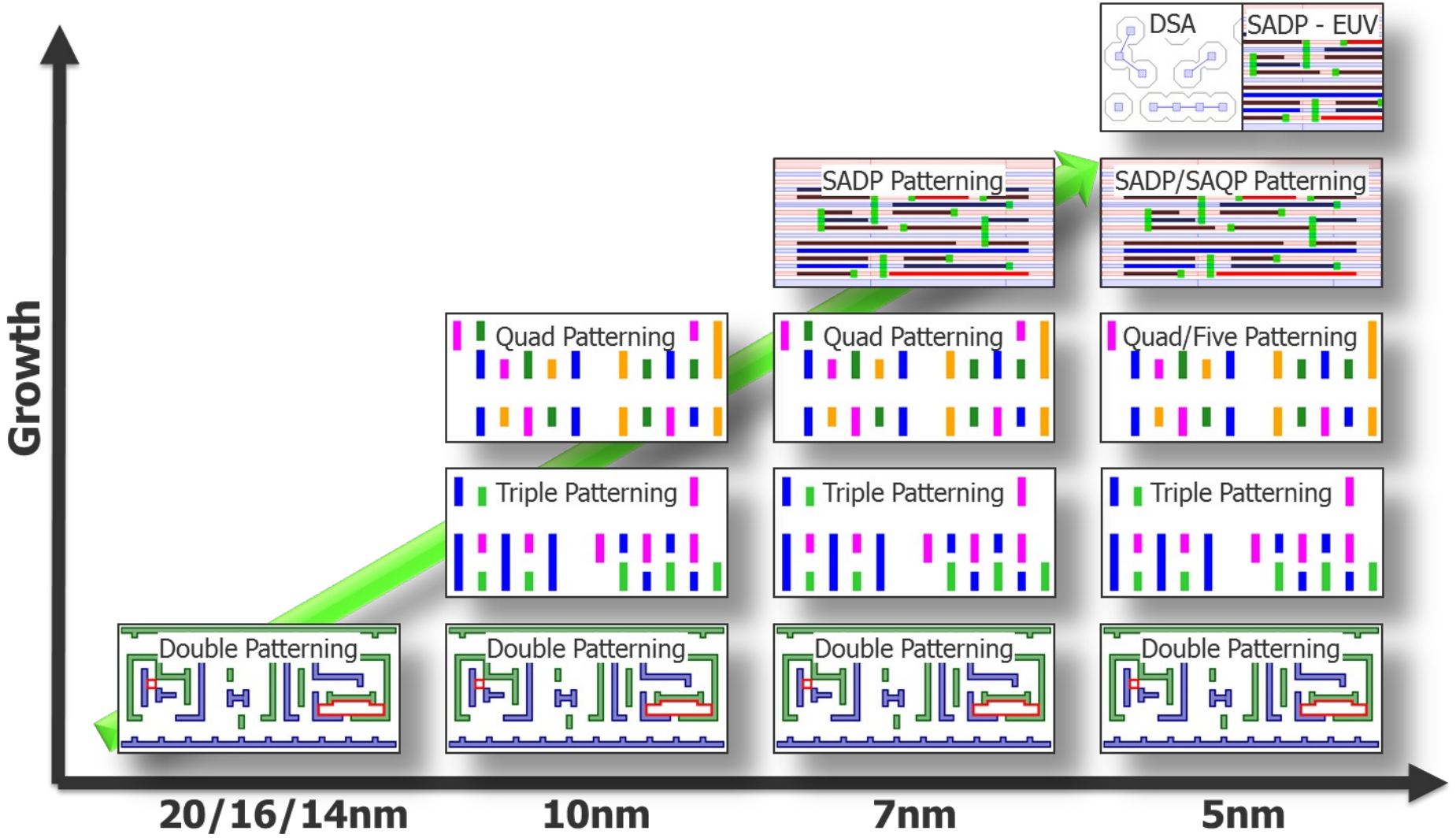
Layout Decomposition for Double Patterning Lithography

Multiple Patterning

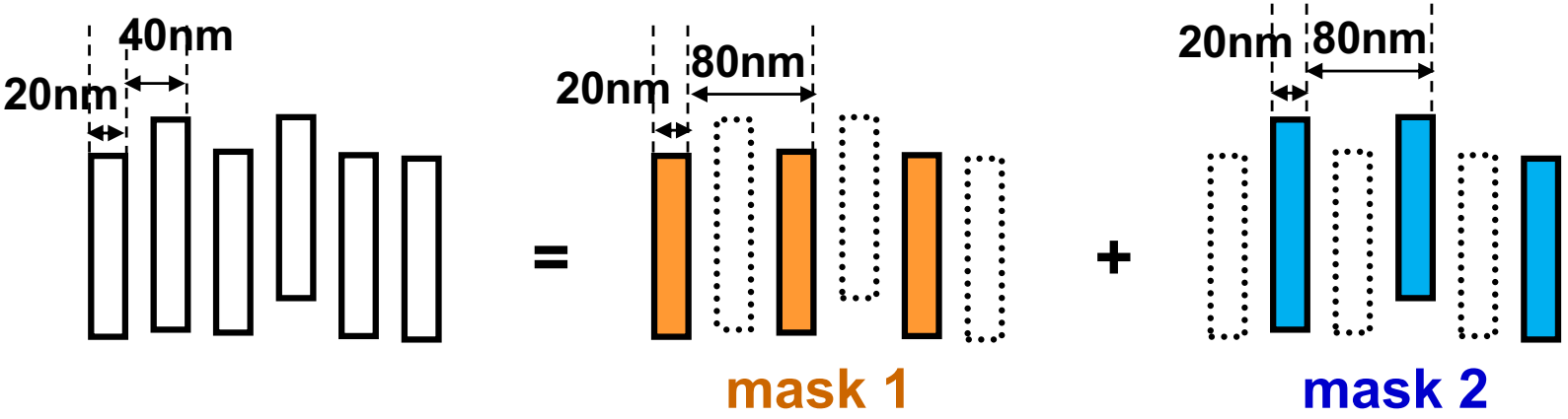
- Technology scaling relies heavily on multiple patterning

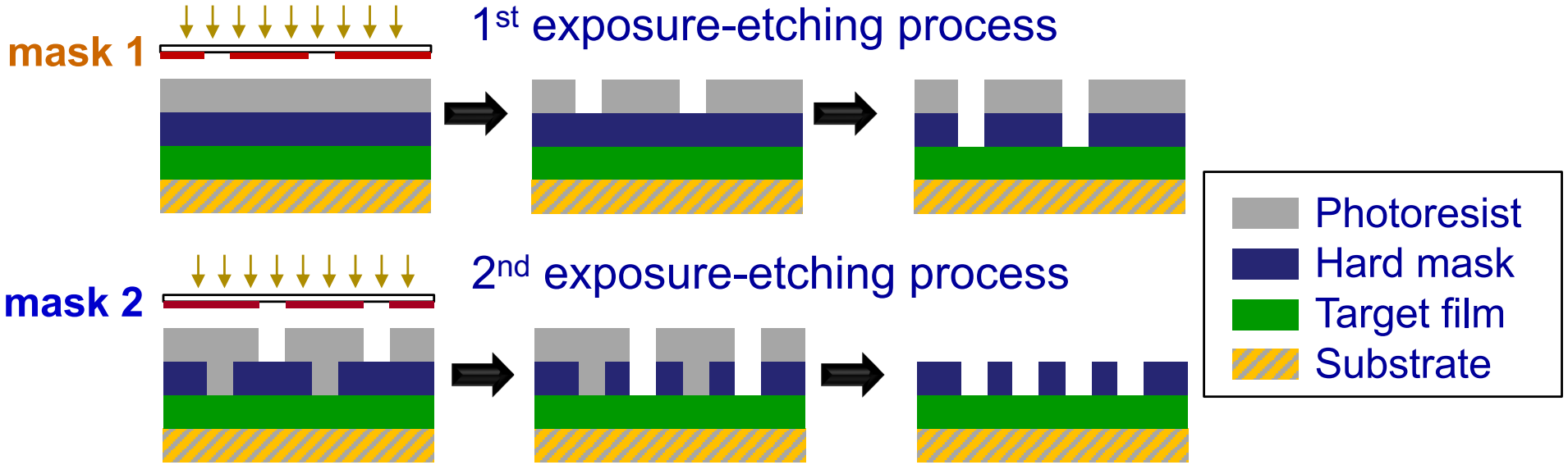
Intel's 10nm Metal Stack			
Layer	**Metal**	**Pitch**	**Patterning**
Fin		34 nm	Quad
Gate	Copper / Cobalt	43-54 nm	Dual
Metal 0	Cobalt	40	Quad
Metal 1	Cobalt	36	Quad
Metal 2, 3, 4	Copper	44	Dual
Metal 5	Copper	52	Dual
Metal 6	Copper	84	Single
Metal 7, 8	Copper	112	Single
Metal 8, 10	Copper	160	Single
Thick Metal 0	Copper	1080	Single
Thick Metal 1	Copper	11000	Single

The Road of Technology Scaling



Litho-Etch-Litho-Etch (LELE) Double Patterning



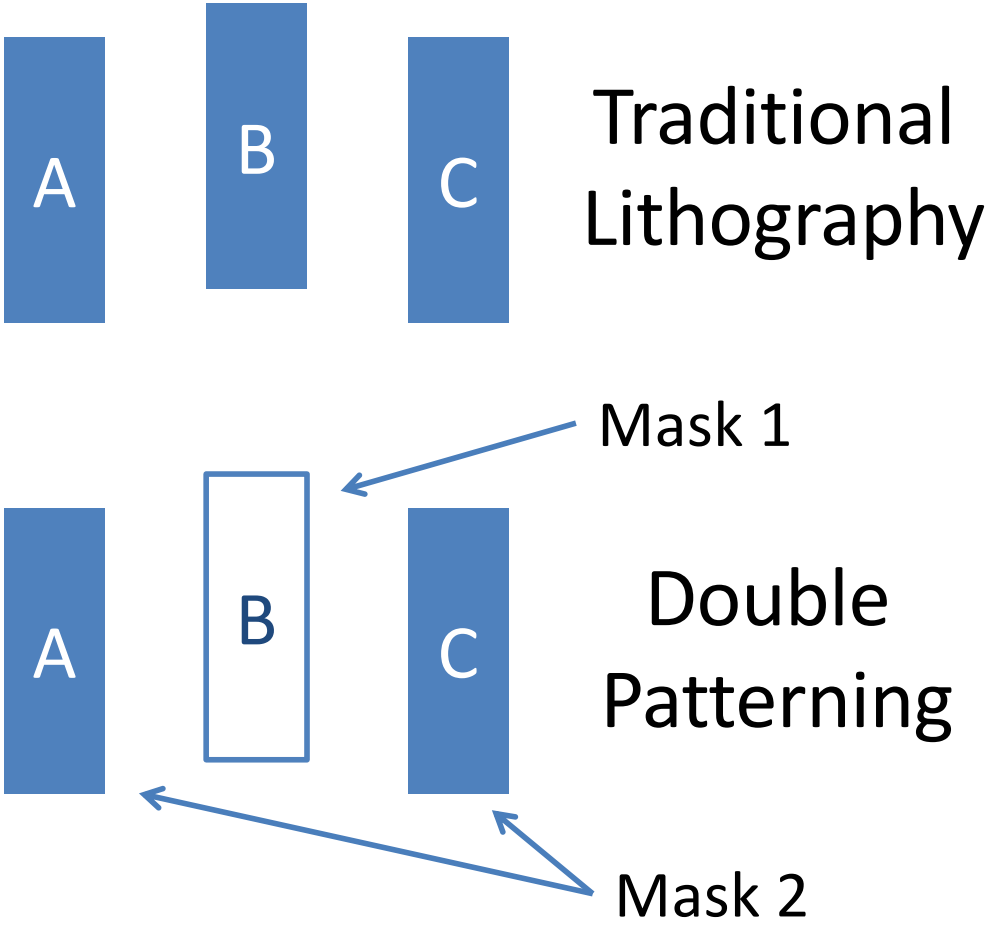


Double Pattering Lithography

- Double Patterning: use two masks and two rounds of pattern transfer to fabricate patterns on one layer.
- Advantage: relax stringent requirement on lithography system by increasing the minimum pitch.
- DPL Conflict: if minimum spacing between two patterns < a predefined threshold, they are conflicting and have to be assigned onto different masks.

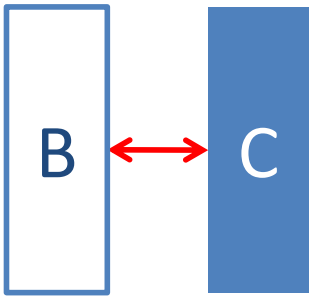
Double Patterning Lithography

- Double Patterning: use two masks

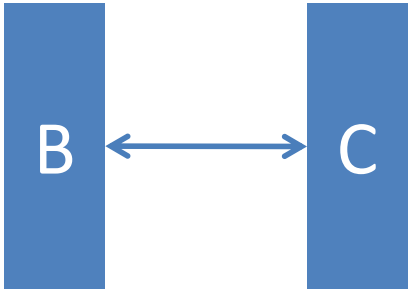


Double Patterning Lithography

- DPL Conflict
  - two very close patterns have to be assigned to different masks



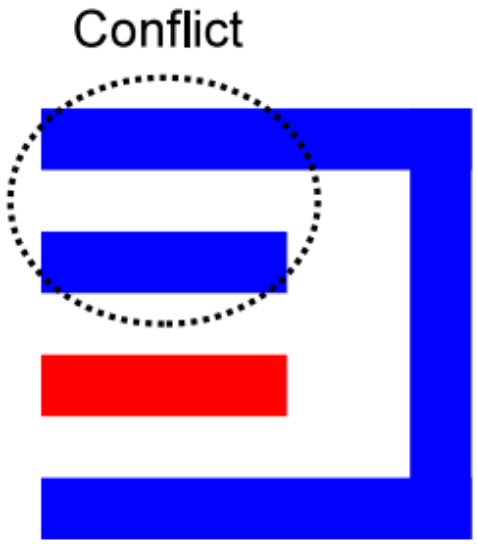
DPL Conflicting



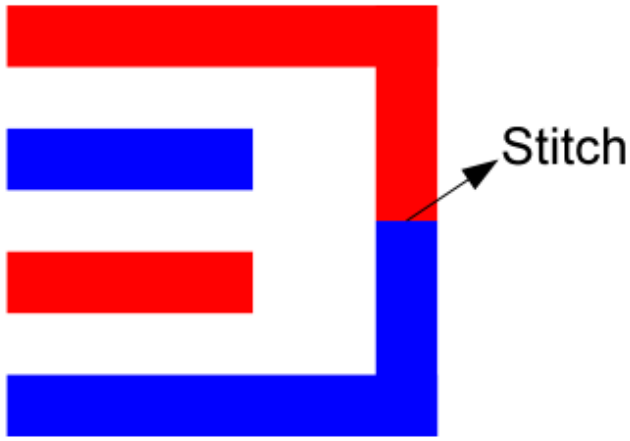
Not DPL Conflicting

Double Patterning Lithography

Target layout

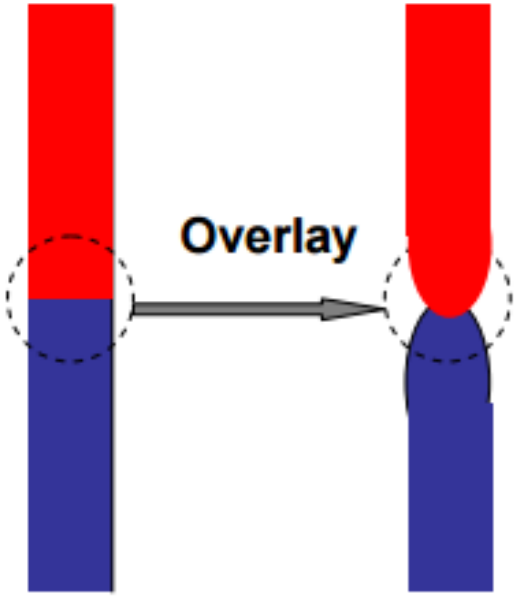


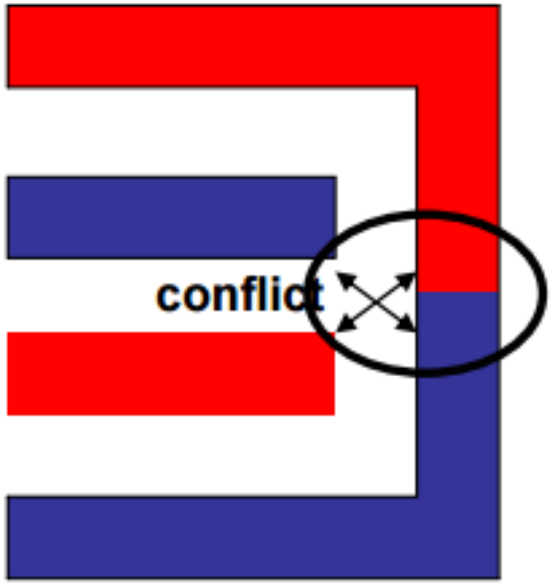
Conflict after layout decomposition



Conflict resolved by stitch insertion

Double Patterning Lithography





Redesign

Stitch reduces yield due to pinching and overlay error

Stitch insertion cannot resolve all conflicts (intrinsic infeasibility)

DPL Decomposition (“Coloring”)

- Objective: To assign patterns on one layer to two masks so that there is no DPT conflict between patterns on each mask
- Pattern slicing and generating stitches are employed to enhance the feasibility of decomposition
- Stitches tend to generate pinching or bridging effects that lower yield so their number should be minimized

GREMA: Graph Reduction Based Efficient Mask Assignment for Double Patterning Technology

Yue Xu and Chris Chu

Outline

- Some Previous Decomposers
- A Graph Reduction Based Decomposer: GREMA
  - Candidate Stitch Generation
  - Graph Reduction Techniques
  - Decomposition Based on Maximum Cut
- Experiment

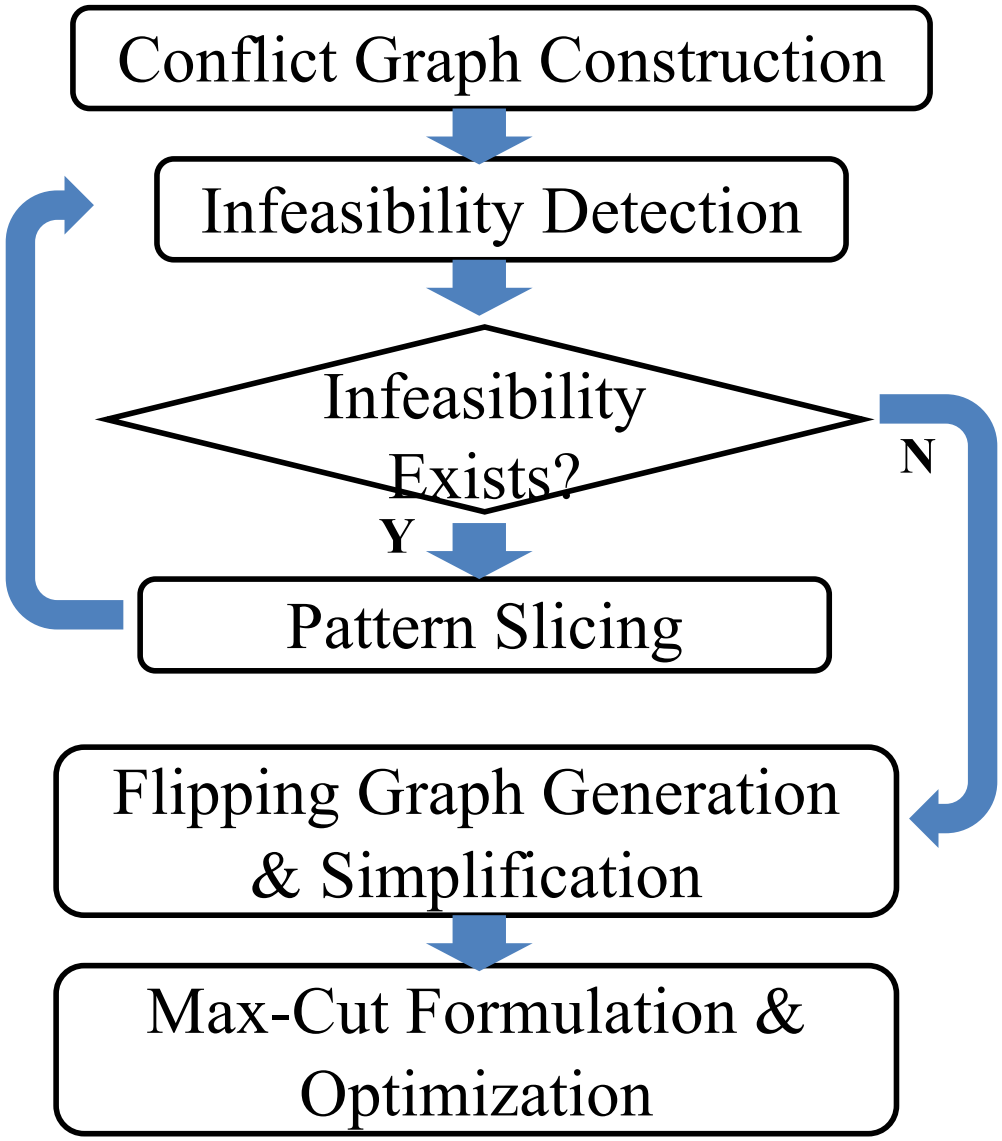
Some Previous Decomposers

- Model Based Decomposer
  - optical simulation
  - too slow for current complex and large-scale layout
- Rule Based Decomposers
  - Heuristics that greedily slice patterns and assign patterns [Kahng+ ICCAD08]
  - Pre-slice layout into fundamental grids and use ILP to select mask assignment for every grid, too slow for large-scale design [Yuan+ ISPD09]

A Graph Reduction Based Decomposer: GREMA [Xu+ ICCAD09]

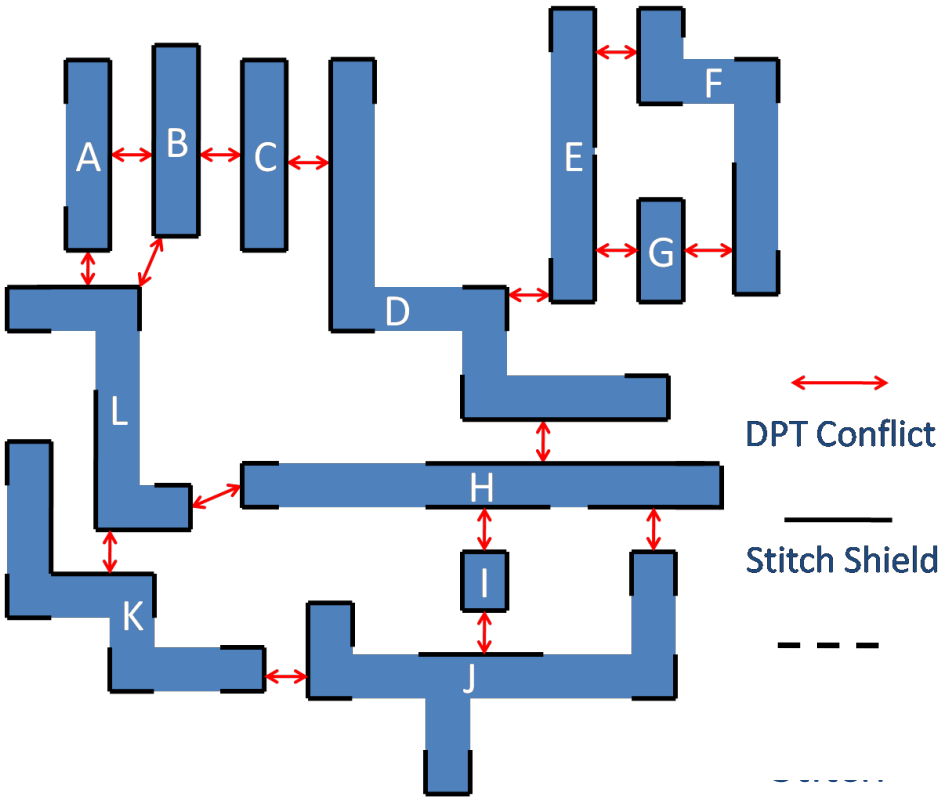
- A two stage scheme that separates generating candidate stitches from minimizing the number of stitches
- Avoid pre-slicing
- Abstract decomposition into a very simple flipping graph
- Flipping graph simplification and a max cut formulation

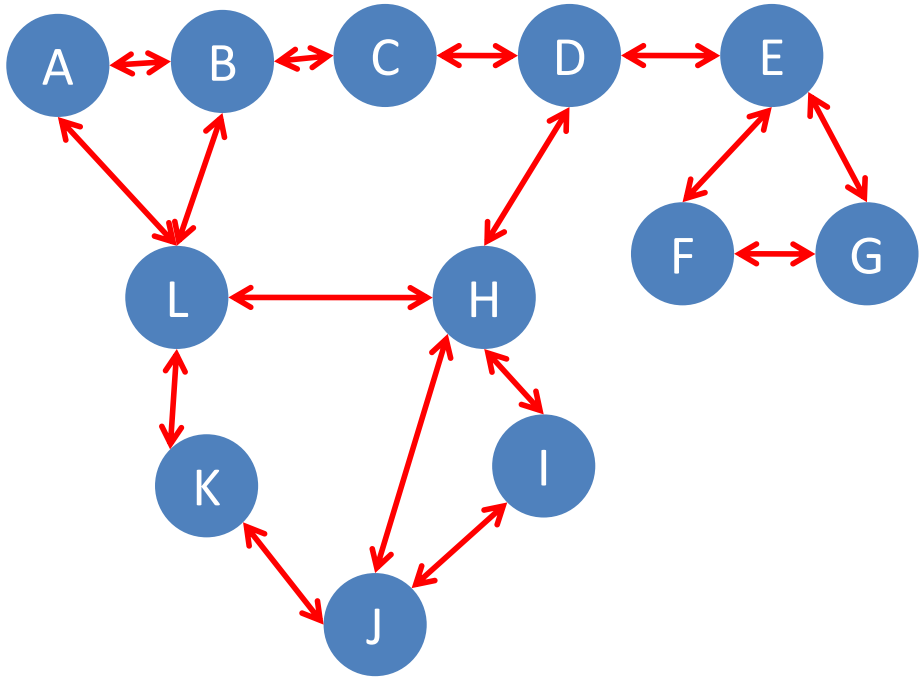
GREMA Flow



N.B. Patterns that need stitches to generate a valid decomposition are represented by odd cycles in Conflict Graph

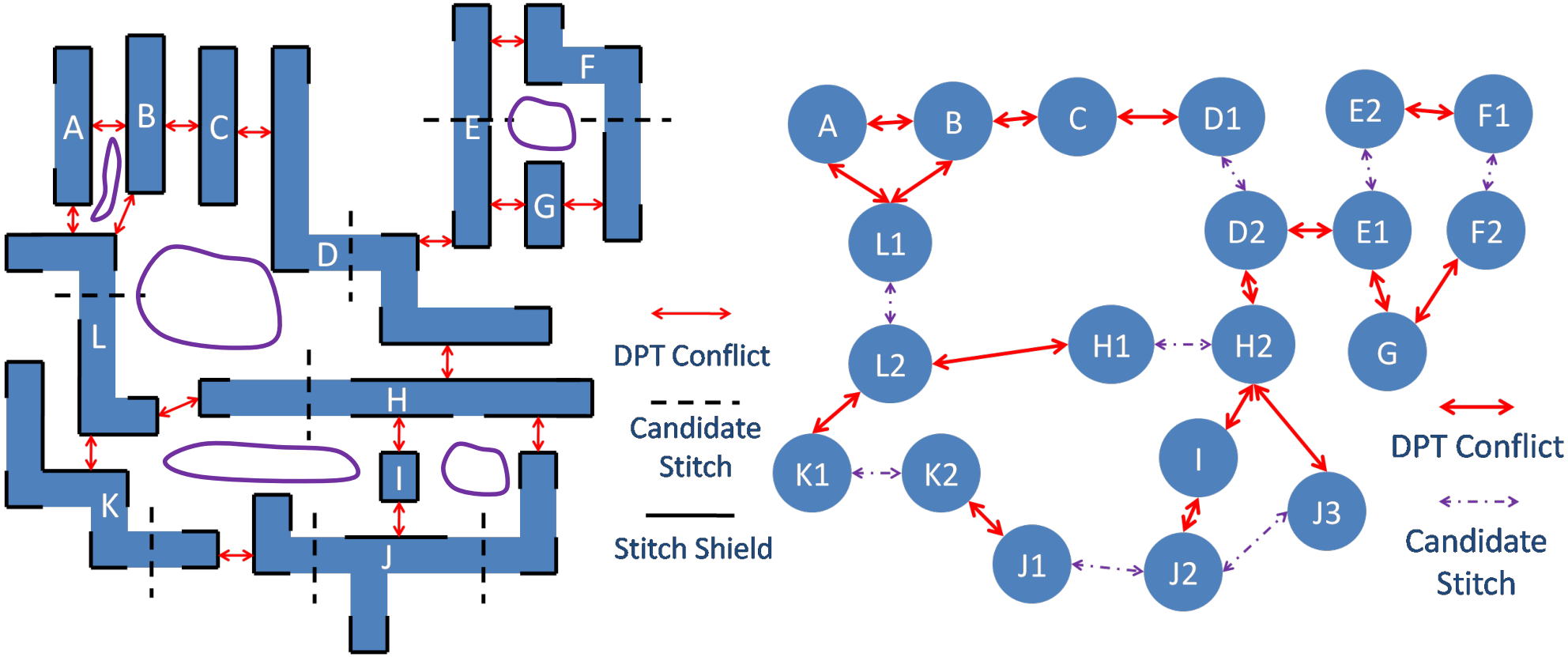
Conflict Graph Construction





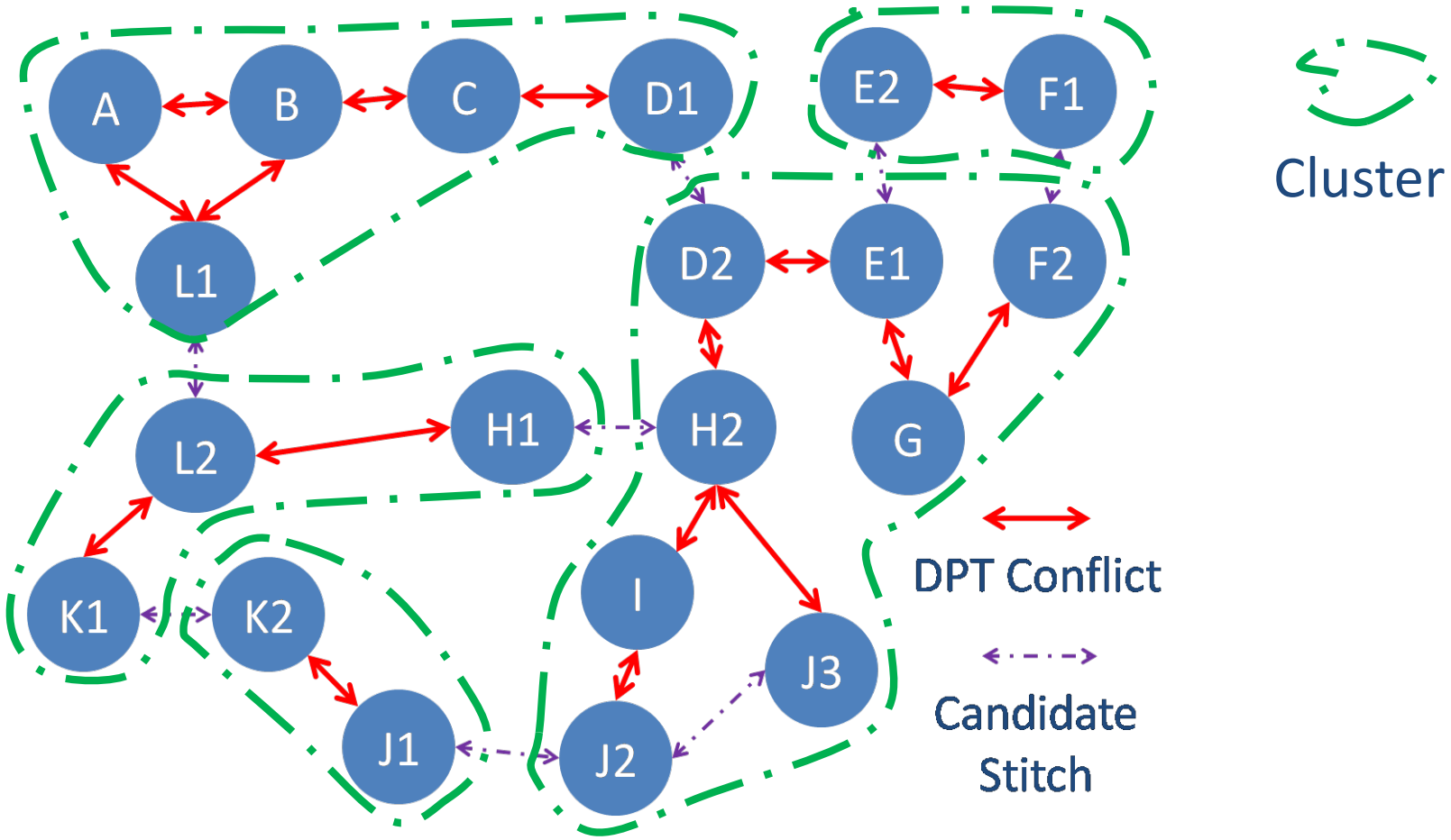
- Odd cycle in conflict graph means no feasible DPT decomposition.

Candidate Stitch Generation and Decomposition Graph Construction



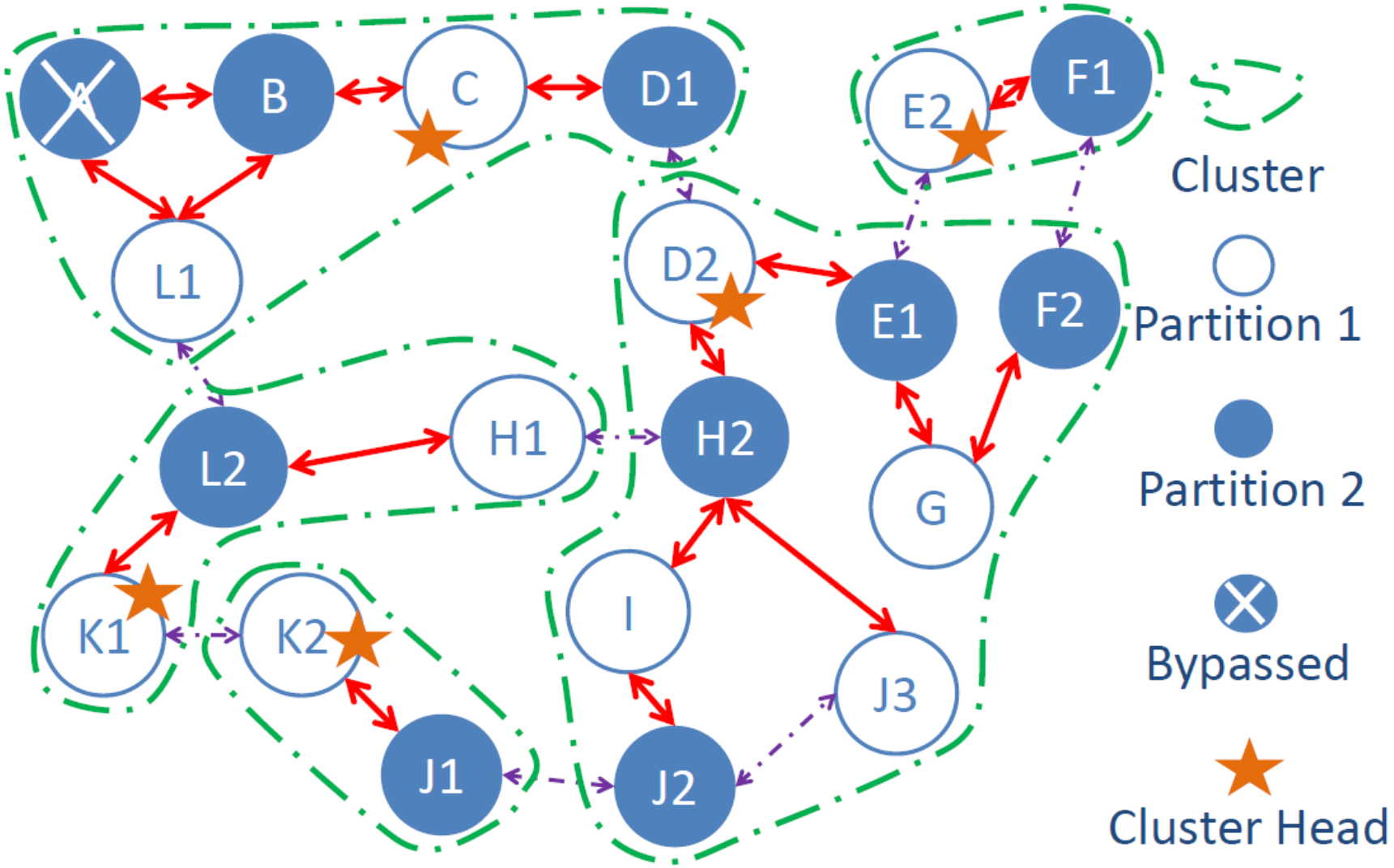
- Only generate useful candidate stitches on patterns in odd cycles.

Cluster Formation

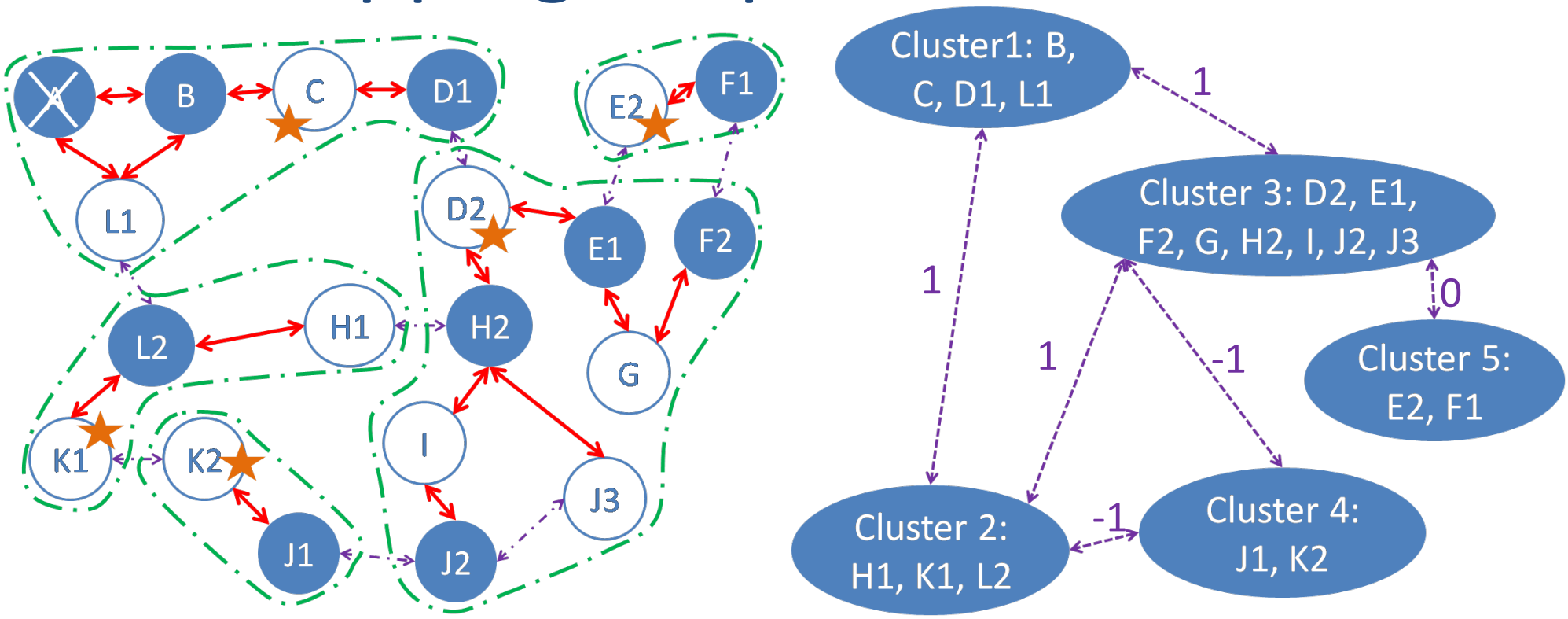


- Form a cluster for each conflict edge connected component.
- Two mask assignment choices for each cluster.

Initial Mask Assignment for Clusters



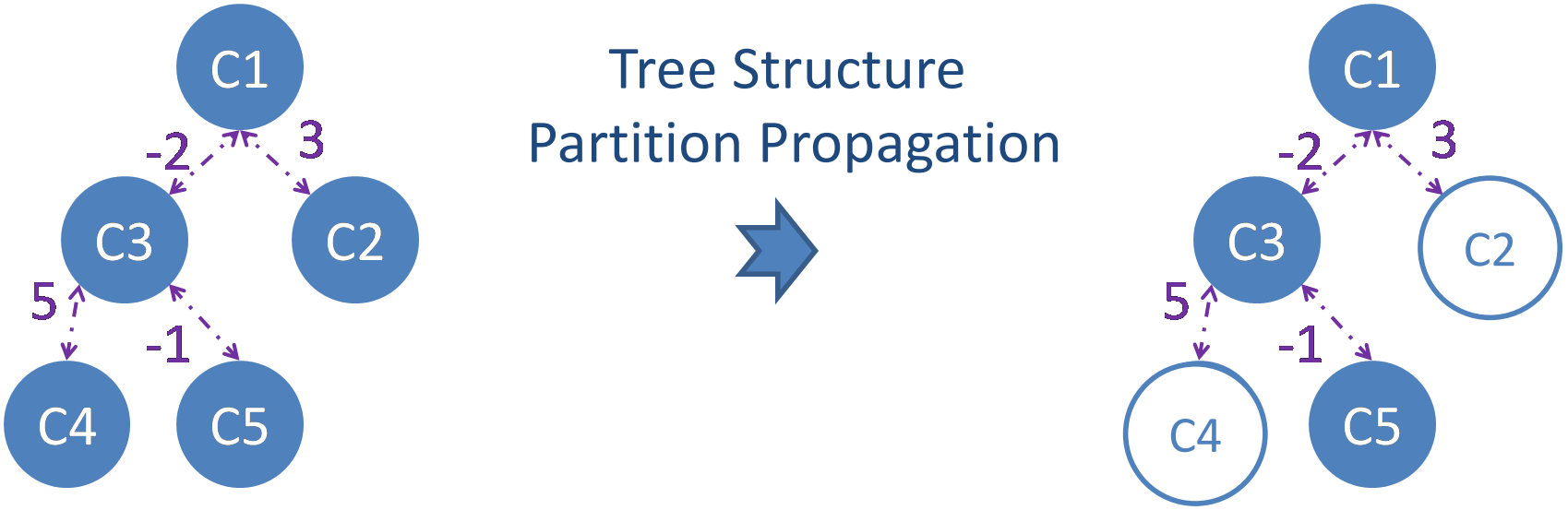
Flipping Graph Generation



- Flipping Graph (FG)
  - gain of edge (i,j) = #stitches reduced by flipping the initial mask assignment of either cluster i or j
- Decomposition is transformed into a maximum cut problem on FG

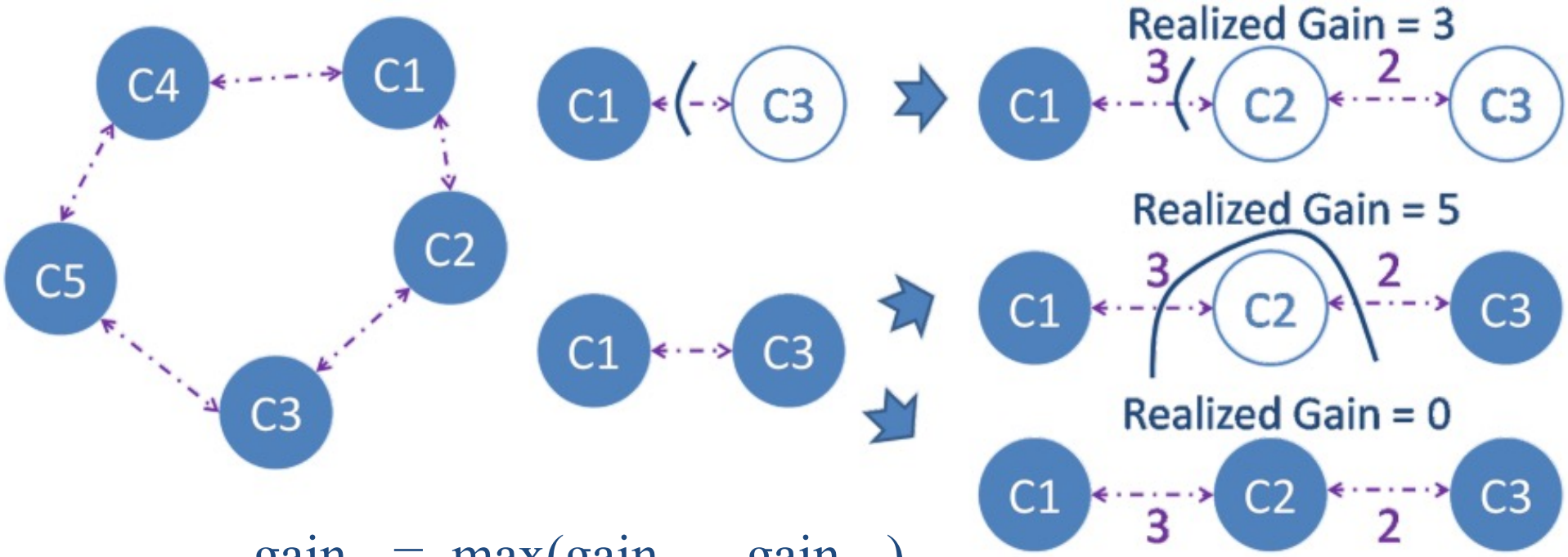
Flipping Graph Simplification

- For tree structures found in flipping graph
  - they can be optimized independently and can be removed from FG



Flipping Graph Simplification

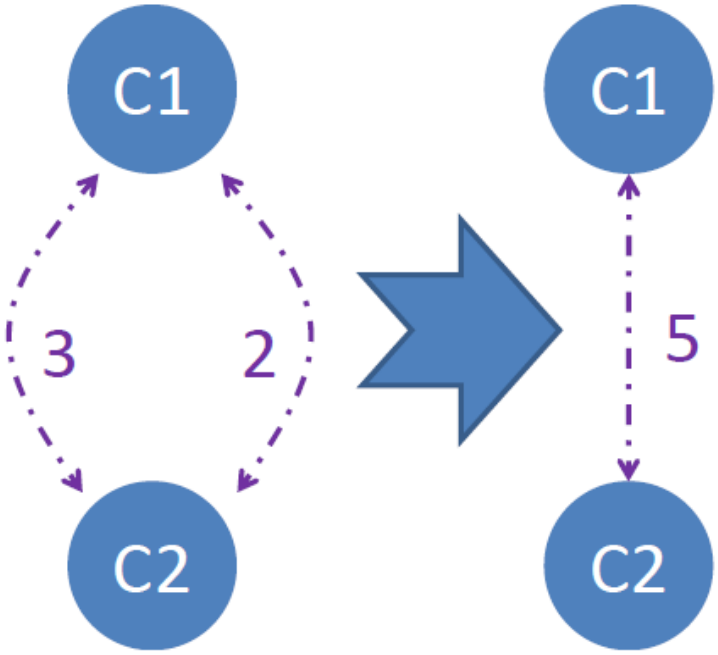
- For 2 serial edges incident to a degree-2 node
  - Serial edge simplification



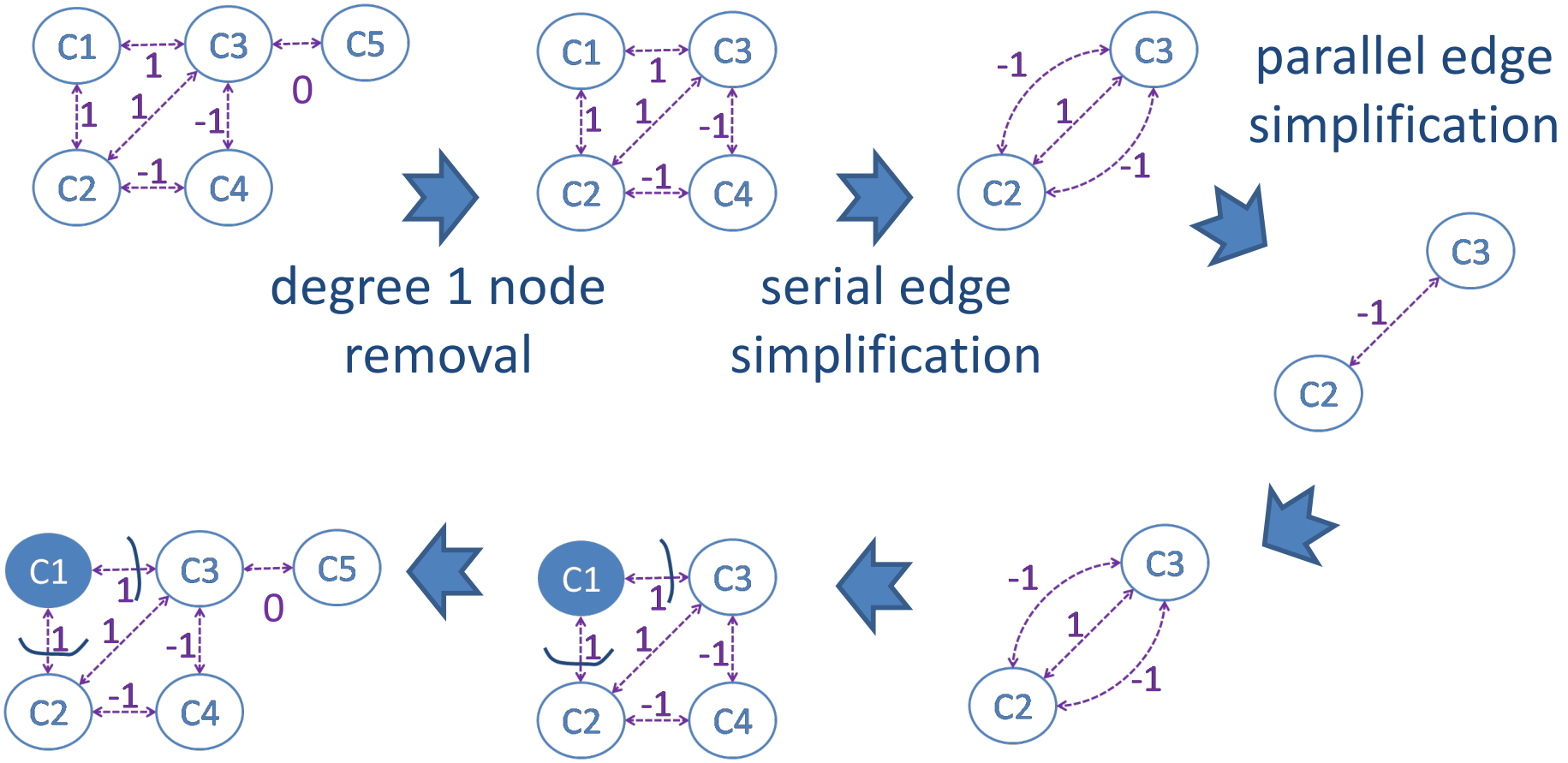
$$gain_{1,3} = \max(gain_{1,2}, gain_{2,3}) - \max(gain_{1,2} + gain_{2,3}, 0)$$
$$= 3 - 5 = -2$$

Flipping Graph Simplification

- For parallel edges
  - Parallel edge simplification

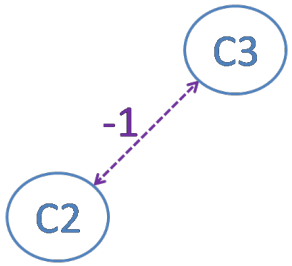


Flipping Graph Simplification Example

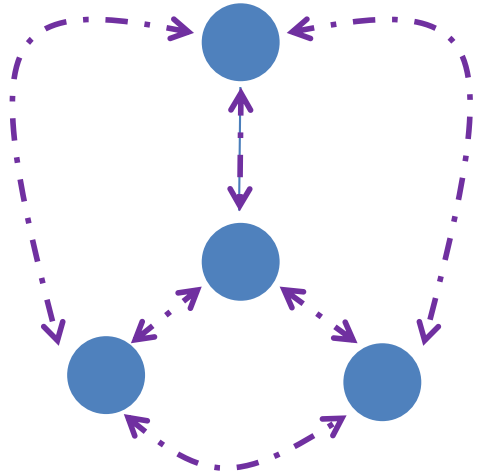


Simplified Flipping Graph

- If final simplified FG just consists of two nodes and one edge, then the flipping decision can be made directly.



- Otherwise, flipping will be decided by finding a max cut in the final FG.



Maximum Cut Problem

- Freedom to flip each cluster in two possible ways.
- The flipping of one cluster will change the realized flipping gain of edges between the cluster and its connected clusters.
- The maximum cut will separates clusters into two partitions and optimally realize the flipping gains of edges in the cut.

The ILP Formulation

$$
\begin{aligned}
OBJ \quad &: \quad max \sum_{i,j} a_{i,j} \\
s.t. \quad &: \quad a_{i,j} \leq f g_{i,j}(\ 2 - (x_i + x_j)) \quad \forall i, j \\
&\qquad a_{i,j} \leq f g_{i,j}\ (x_i + x_j) \qquad\quad \forall i, j \\
&\qquad x_i \ \in \ \{0, 1\} \quad \forall i
\end{aligned}
$$

- x_i : The partition the i^{th} cluster head belongs to
- $f_{i,j}$: Flipping gain between cluster C_i and C_j
- $a_{i,j}$: Realized flipping gain between cluster C_i and C_j

Experiment Setup and Result

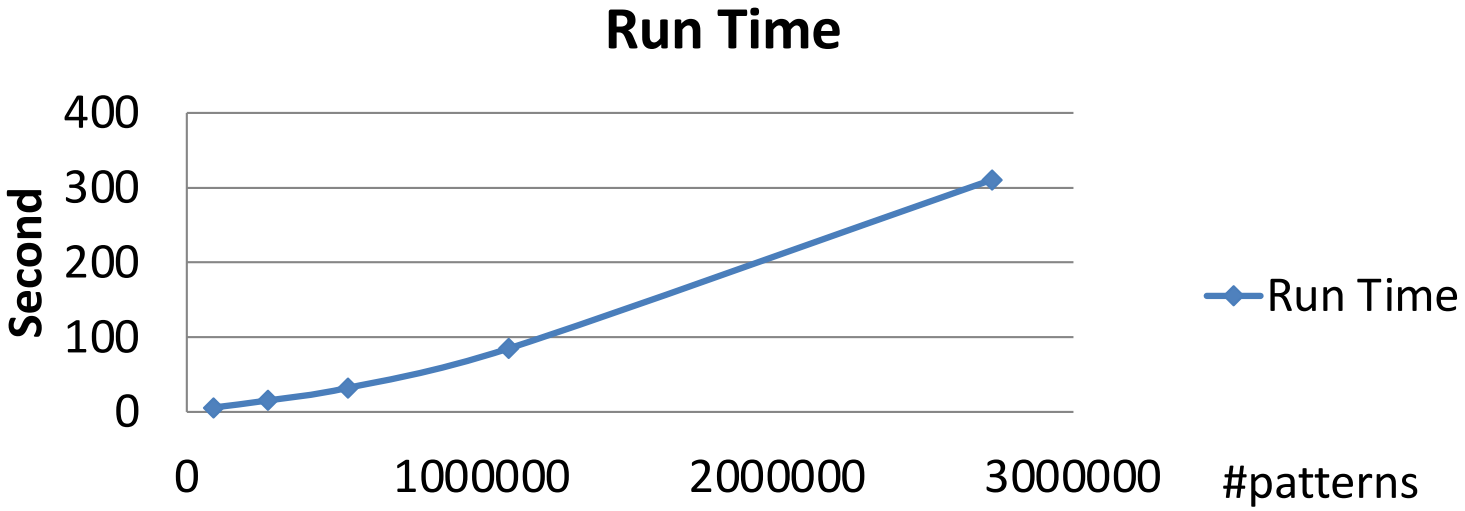
- GREMA is programmed in C
- Simulation is run on a 2.8GHz Intel Xeon Linux machine with 4GB RAM

		GREMA		[Kahng+ ICCAD08]	
Design	# Patterns	# Stitches	CPU time (s)	# Stitches	CPU time (s)
AES	90394	30	5.9	33	17.8
TOP-A	275650	6191	15.4	7903	155
TOP-B	545000	10265	32.2	15755	500.2
TOP-C	2725000	64385	310.7	78709	4655
TOP-D	1090000	24767	85.0		

Graph Reduction Effects

Avg. #nodes per connected component
(CG: conflict graph, DG: decomposition graph, FG: flipping graph)

Design	CG -> DG	DG->FG	FG Simplification
AES	2.23 -> 5.61	5.61 -> 2.13	2.13 -> 2.00
TOP-A	4.28 -> 13.89	13.89 -> 4.81	4.81 -> 2.08
TOP-B	4.26 -> 13.73	13.73 -> 4.66	4.66 -> 2.05
TOP-C	4.26 -> 14.02	14.02 -> 4.85	4.85 -> 2.08
TOP-D	4.27 -> 13.77	13.77 -> 4.79	4.79 -> 2.06



Conclusions

- A graph-reduction based DPT decomposer
- Maximum Cut based formulation of DPT decomposition
- Techniques to make assignment decisions before final ILP based decision
- GREMA generates less stitches for a decomposition while using much less runtime

References

- [Kahng+ ICCAD08] Layout decomposition for double patterning lithography
- [Yuan+ ISPD09] Double patterning layout decomposition for simultaneous conflict and stitch minimization
- [Xu+ ICCAD09] GREMA: Graph Reduction Based Efficient Mask Assignment for Double Patterning Technology

A Matching Based Decomposer for Double Patterning Lithography

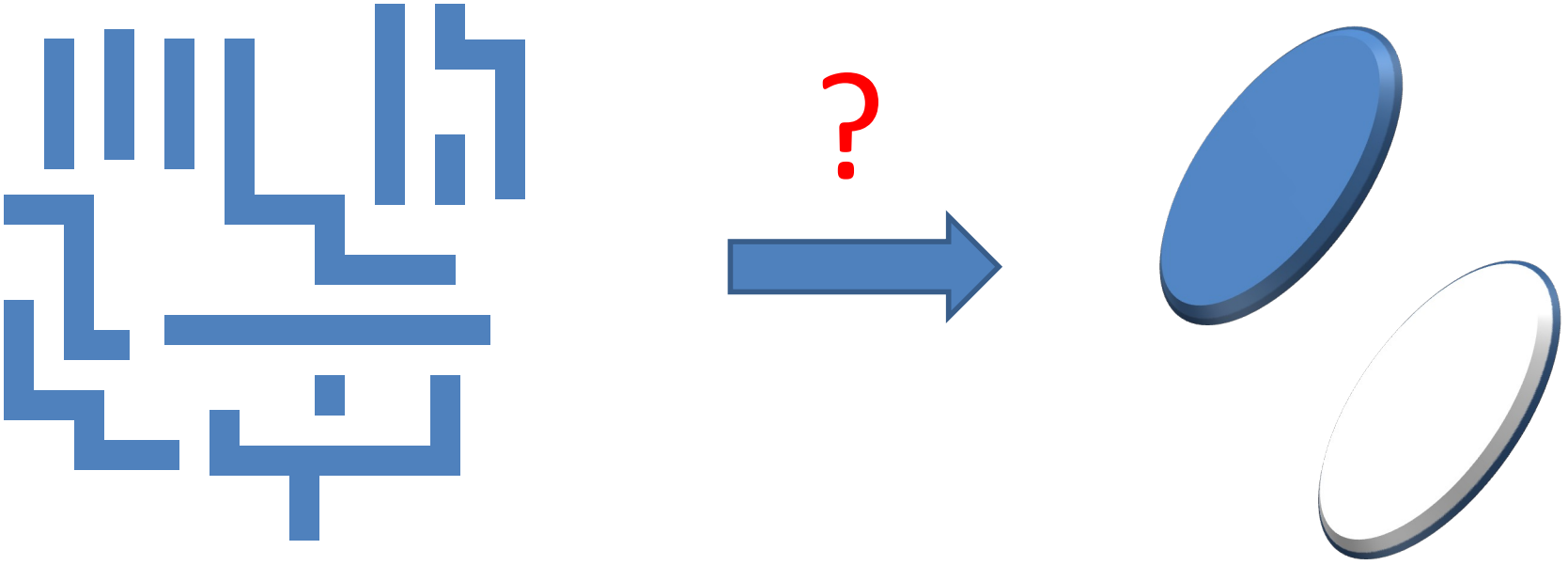
Yue Xu and Chris Chu

Outline

- Planarity Proof of Conflict Graph
- Matching Based Decomposition Algorithm
- Experiment

DPL Decomposition Problem

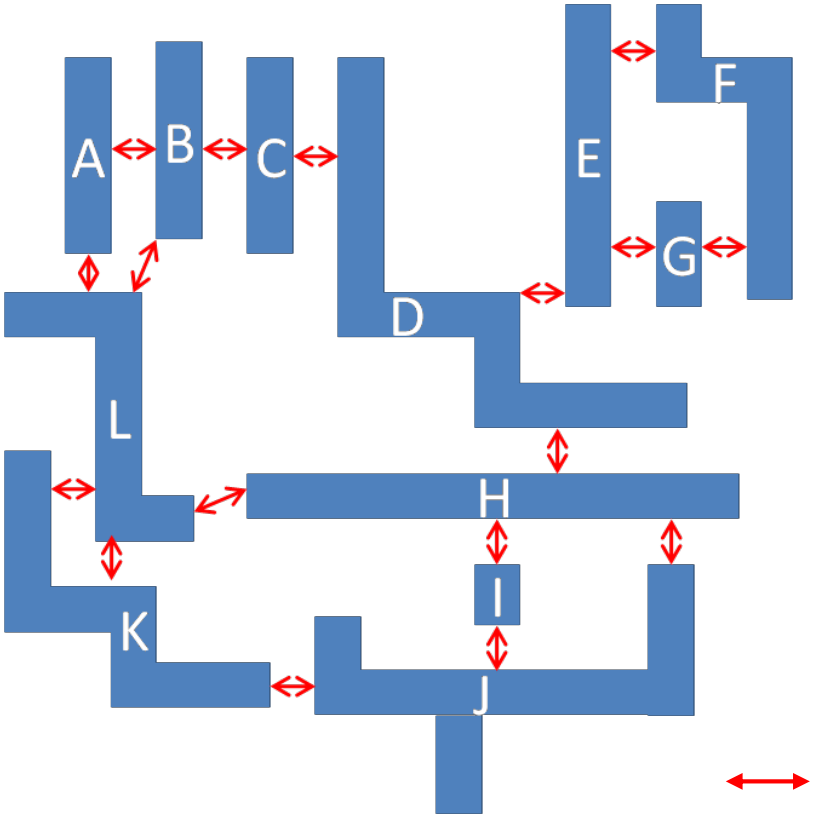
- Objective: To assign patterns on one layer to two masks and resolve every non-intrinsic infeasibility with minimum number of stitches

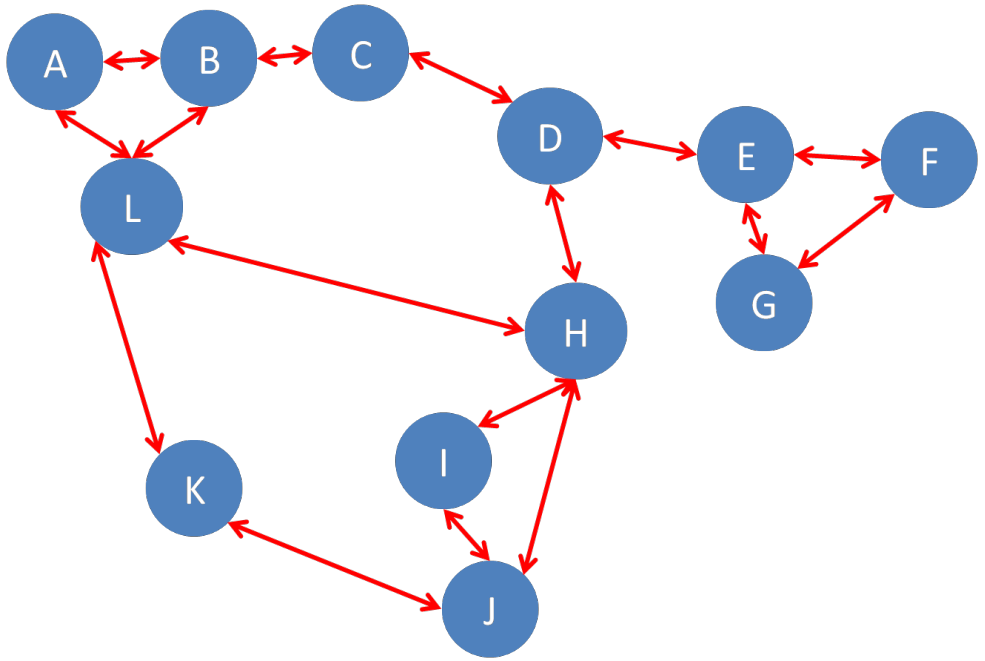


This Work

- Prove that Conflict Graph is planar
- Create Face Graph to model Conflict Graph's odd cycle removal operations
- Propose a new framework for optimal DPL decomposition
- Reduce odd-node pairing problem in the entire FG into a set of sub-problems
- Transform the pairing problem into minimum weighted matching problem
- Use an polynomial runtime maximum weighted matching to solve the minimum weighted matching

Conflict Graph





DPL conflict

Layout

Conflict graph

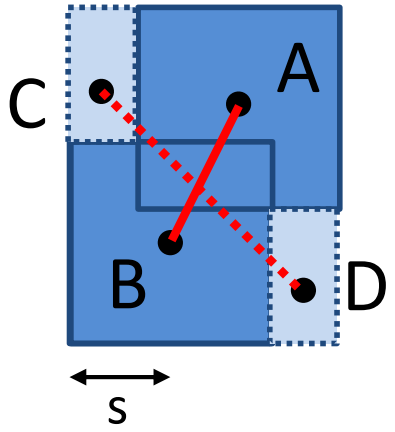
Planarity Of Conflict Graph

Lemma 1: Conflict graph is planar, i.e., can be drawn on a plane without crossing edges, if DPL threshold t $\leq 2 \cdot$ min spacing.

- We will show a stronger result that even if all conflict edges are drawn as straight lines, the conflict graph can still be drawn on a plane without crossing edges.

Planarity Of Conflict Graph

Proof. We want show that if two points A and B are in conflict, then no points in the light shaded regions can be in conflict. Let C and D be two points in the light shaded regions.



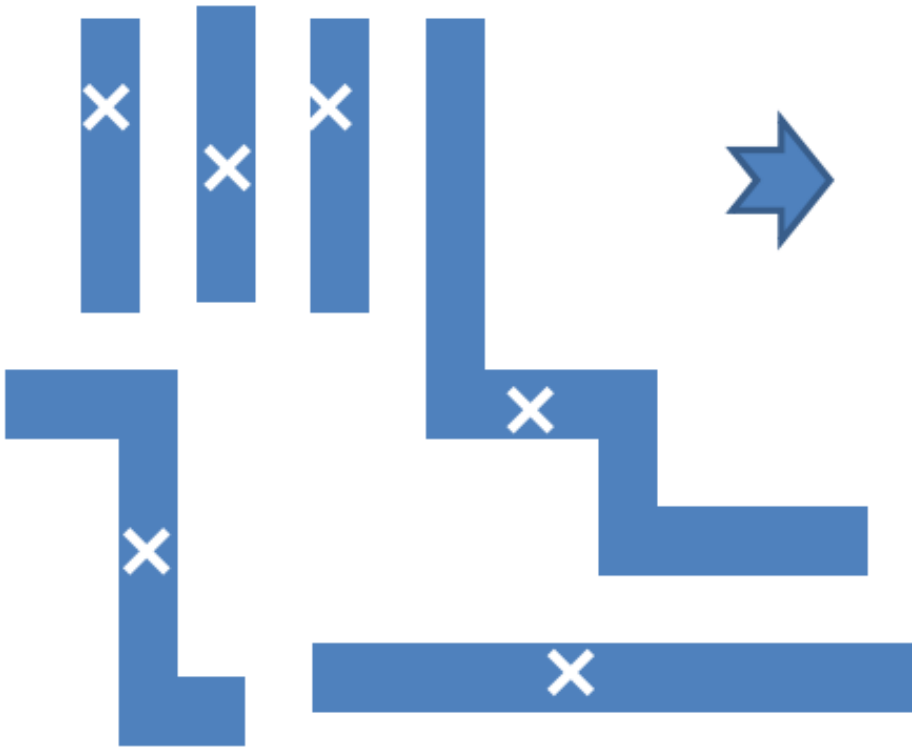
We have $x_A - x_B + y_A - y_B < t$

$$x_C \leq x_A - s$$
$$y_C \geq y_B + s$$
$$x_D \geq x_B + s$$
$$y_D \leq y_A - s$$

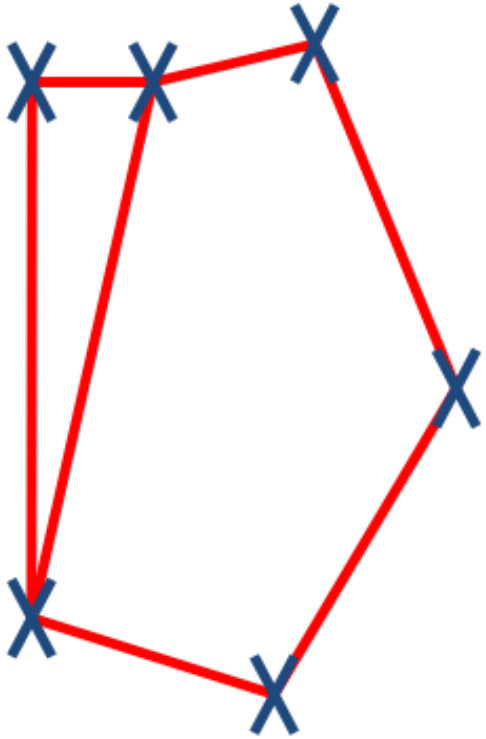
$$\begin{aligned} MD(C,D) &= x_D - x_C + y_C - y_D \\ &\geq x_B - x_A + 2s + y_B - y_A + 2s \\ &> 4s - t \end{aligned}$$

Since $t \leq 2s$, $MD(C,D) > 2s \geq t$

Planar Embedding

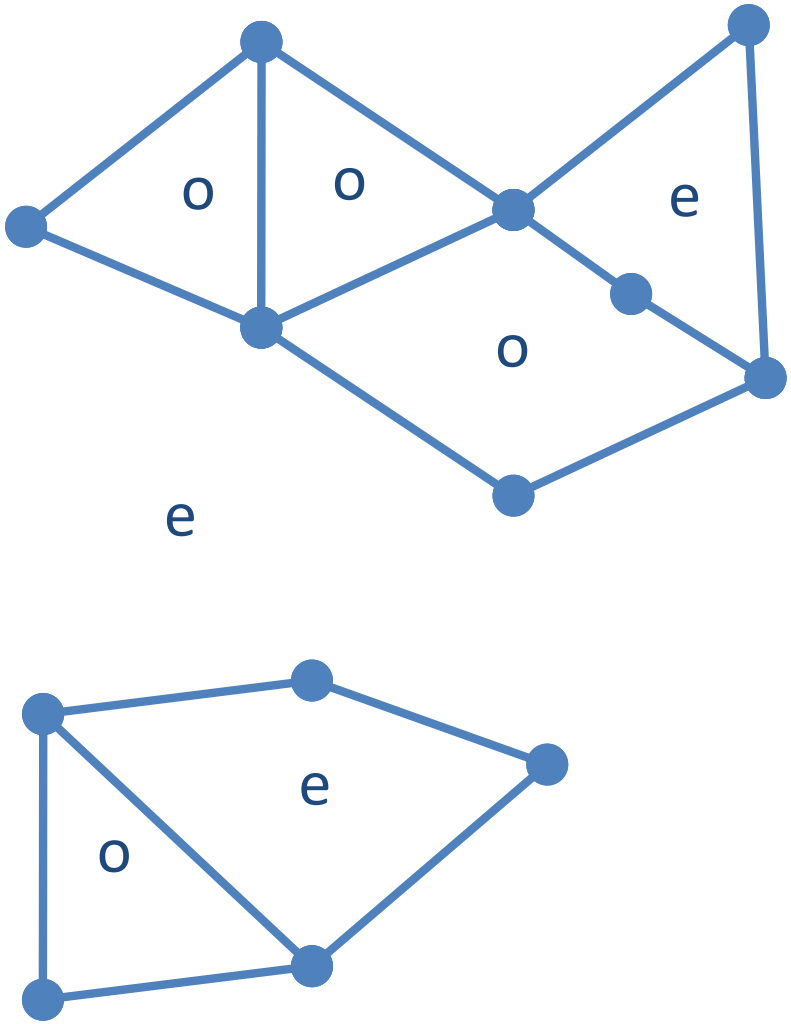


Layout



Planar representation of conflict graph

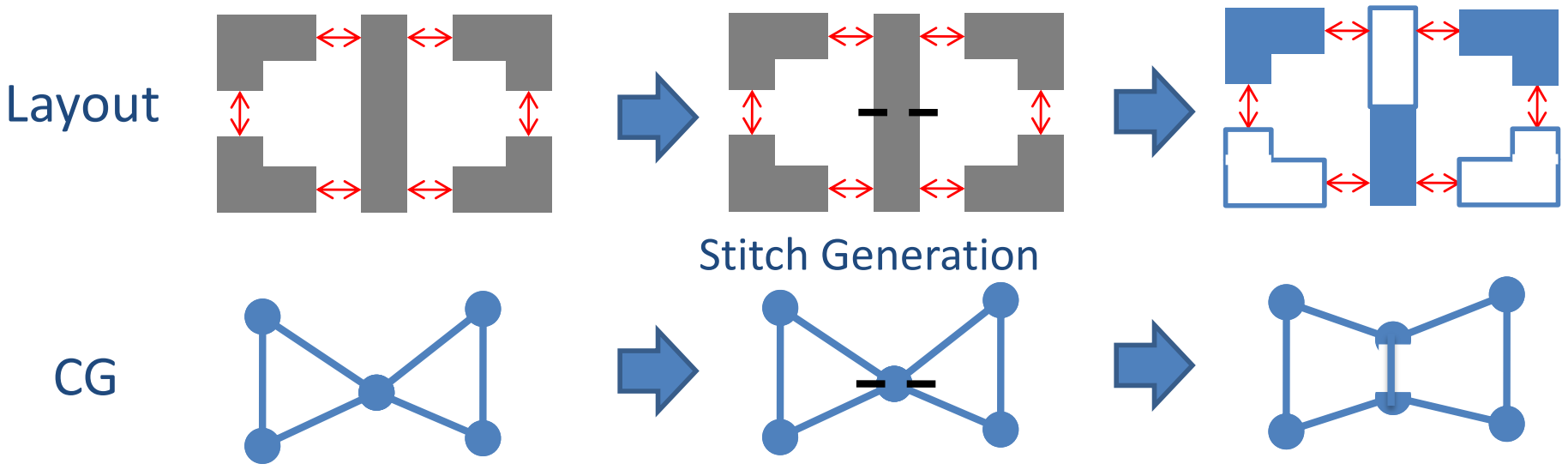
Odd Cycles



- Odd cycles in conflict graph CG obstruct DPL decomposition
- How to eliminate all odd faces in CG?
  - Stitch generation
  - Layout modification

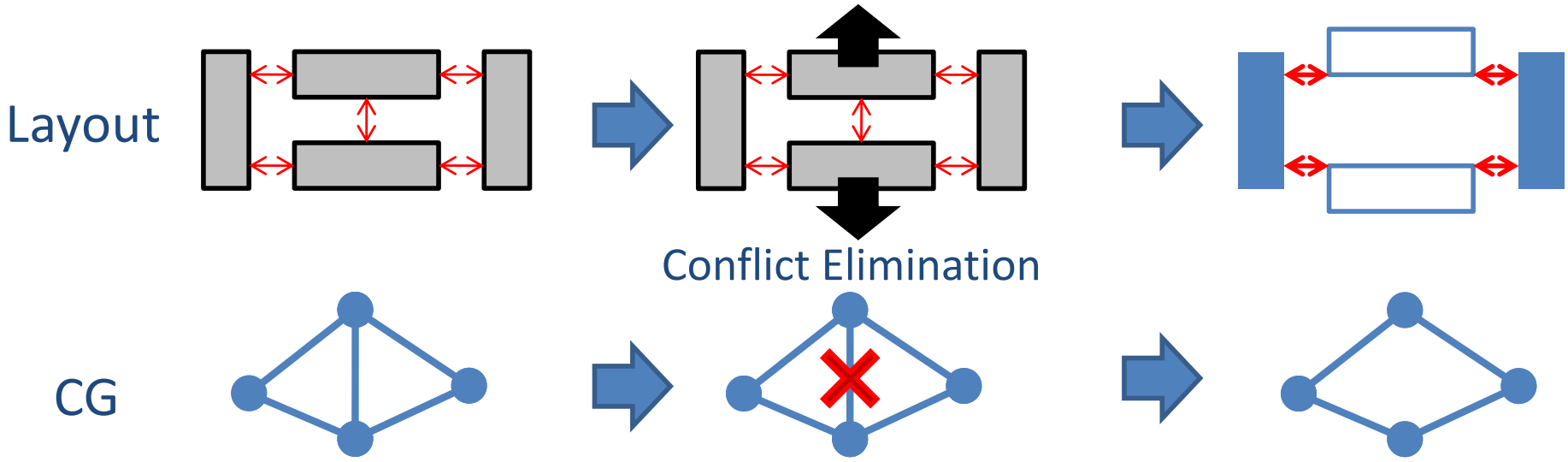
Stitch Generation: Node Splitting

- Stitch generation
  - correspond to node splitting in conflict graph
  - increase #boundaries of two neighboring faces by 1



Layout Modification: Edge Removal

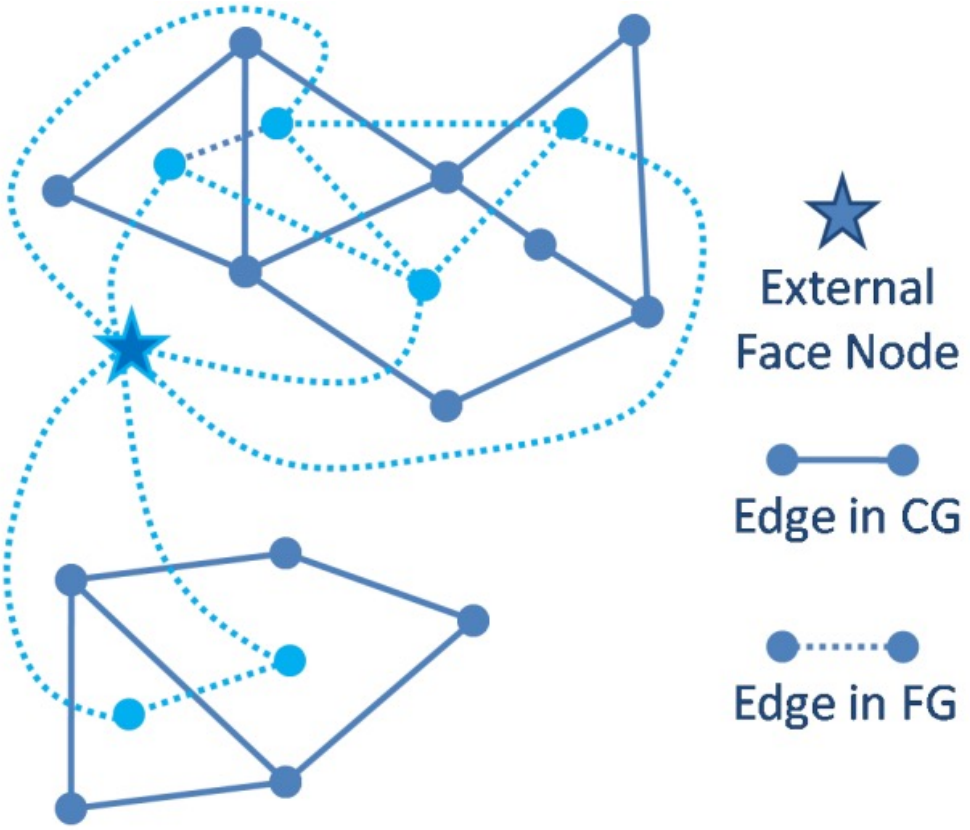
- Layout modification
  - correspond to edge removal in conflict graph
  - cause two neighboring faces to merge



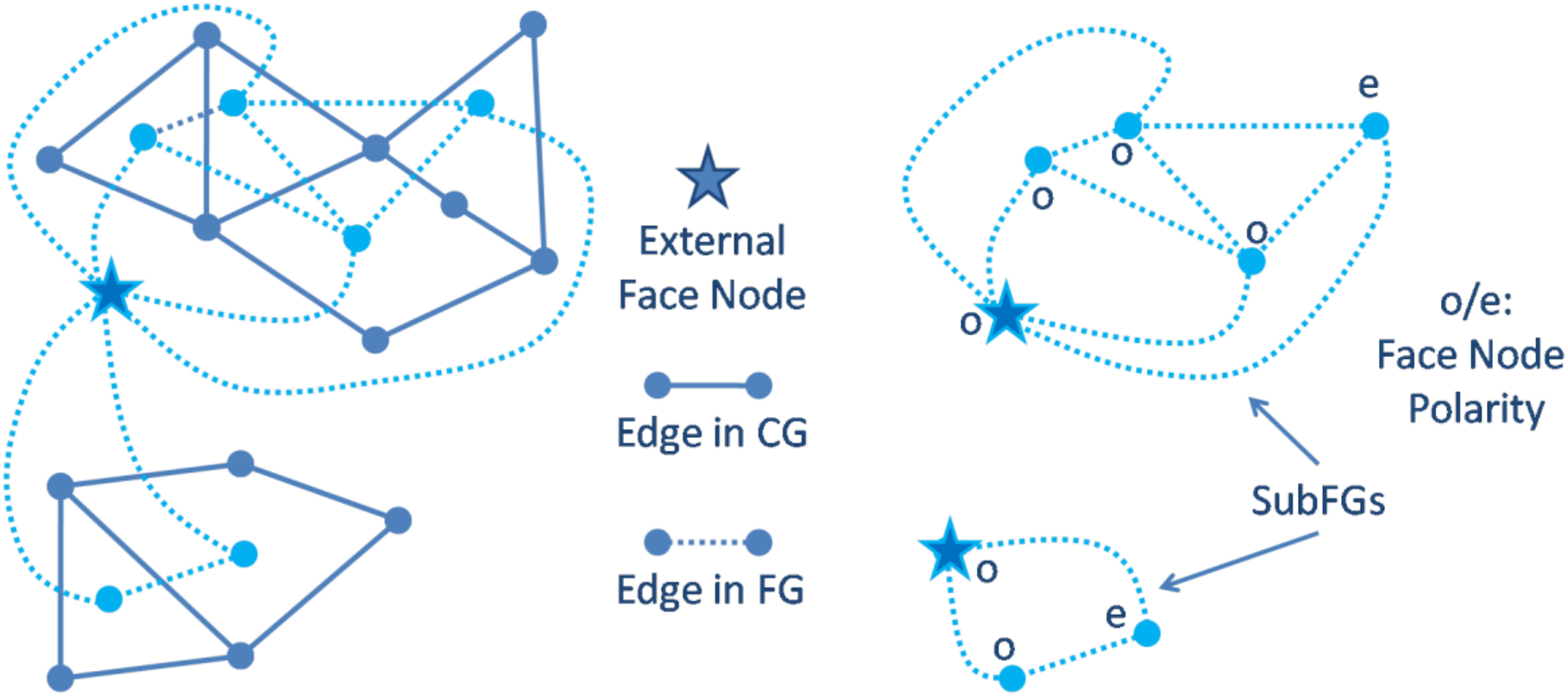
Face Graph Construction

Face Graph (FG)

- each node represents a face in CG, including the external face
- an edge with weight W_s connects two nodes f1 and f2 if we can generate a stitch that increases the #boundaries of both faces f1 and f2 by 1
- an edge with weight W_e connects two nodes f1 and f2 if we can merge faces f1 and f2 by edge removal



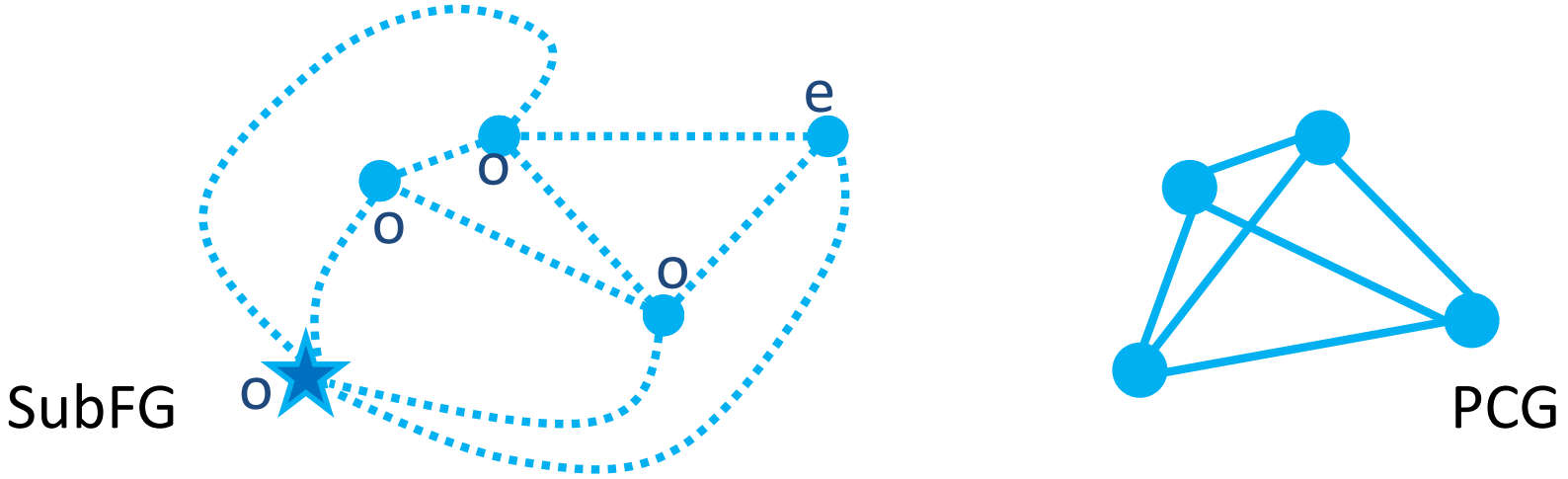
Face Graph Partitions



One SubFG per connected component in CG

SubFG Simplification

- Pair up odd faces in each SubFG, using even faces as agents if needed
- Use Floyd-Washall algorithm to find the shortest path for each pair of odd nodes
- Create a complete graph called Pairing Cost Graph (PCG) on the set of all odd nodes
  - Pairing cost = shortest path cost in the original SubFG

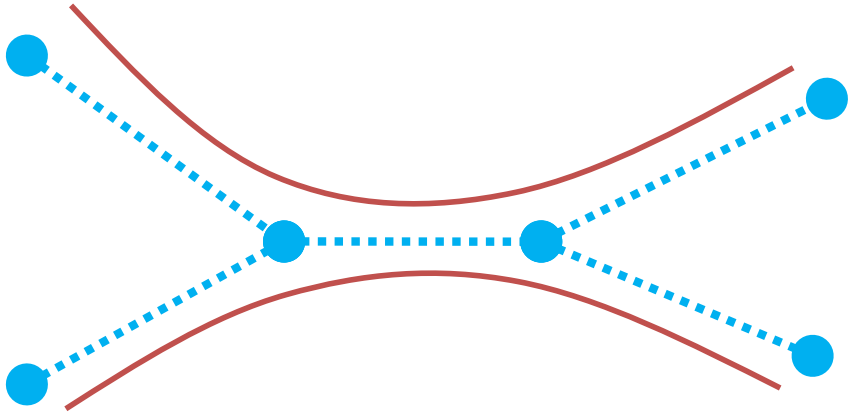


Matching Based Solution

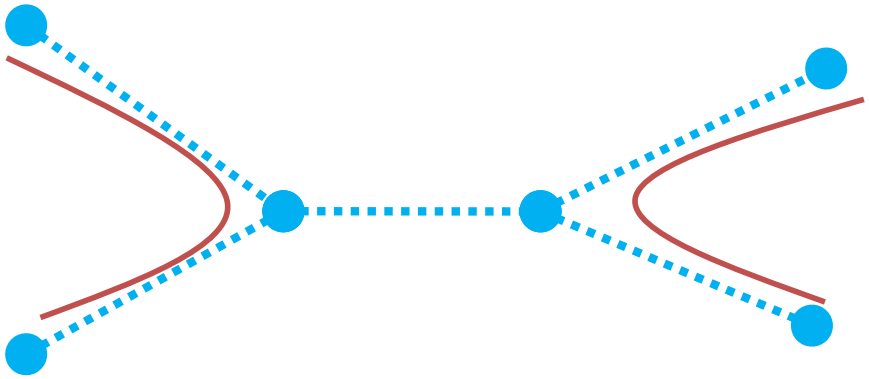
- The odd node pairing in the simplified SubFG is a minimum weighted perfect matching problem
- Convert the minimum weighted perfect matching problem by changing edge weight into a maximum weighted matching problem

Remark

- The minimum weighted matching solution found in Pairing Cost Graph (PCG) will never use an edge in the face graph (FG) twice

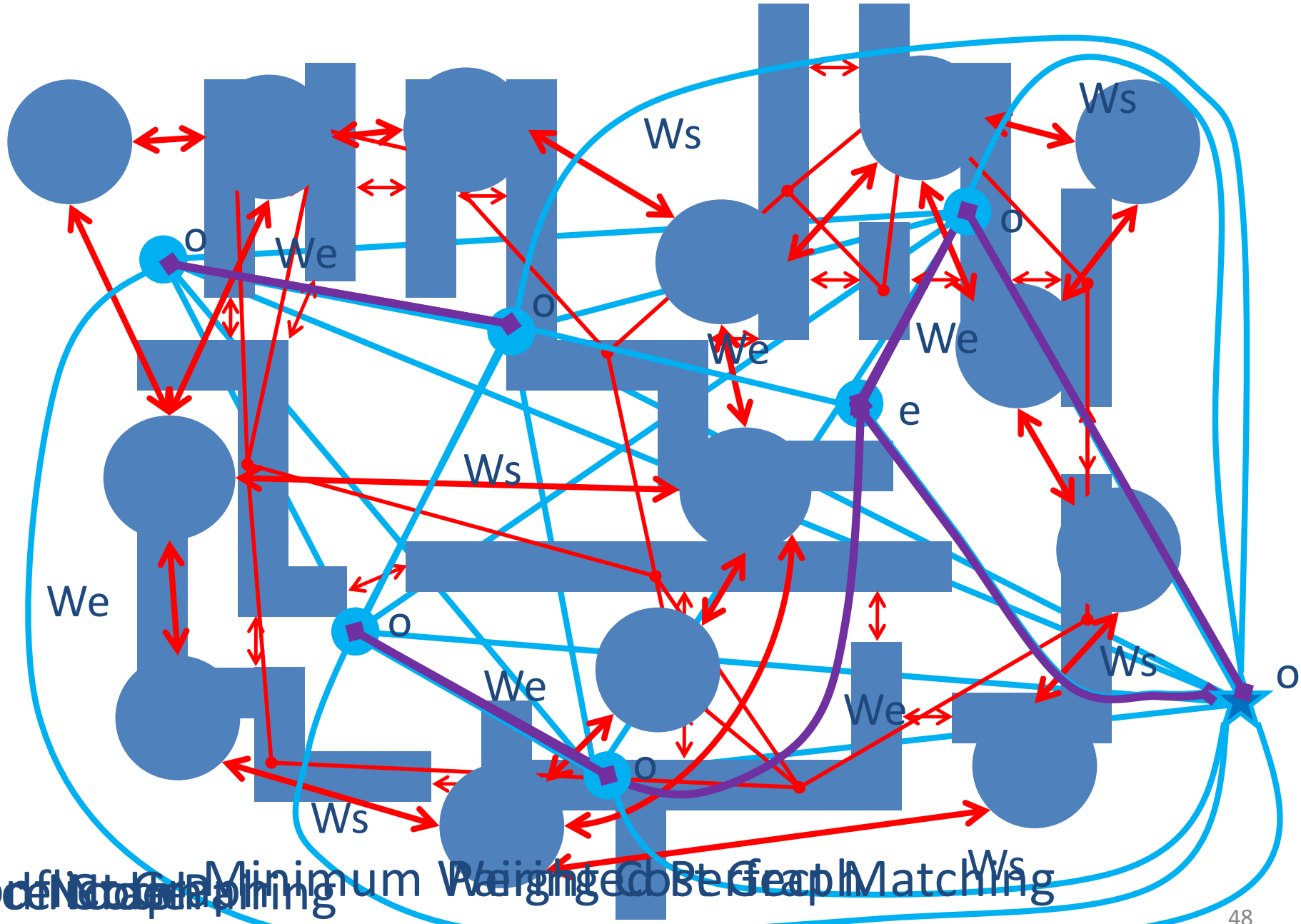


Pairing of two odd node pairs in FG that share an edge



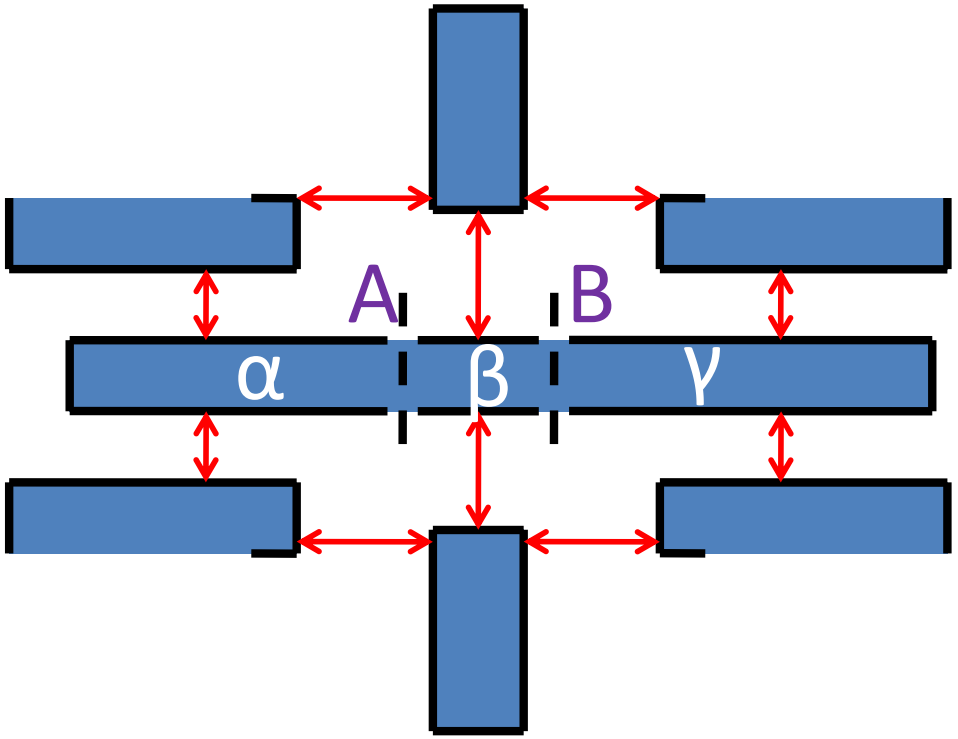
A better pairing for the same two odd node pairs in FG

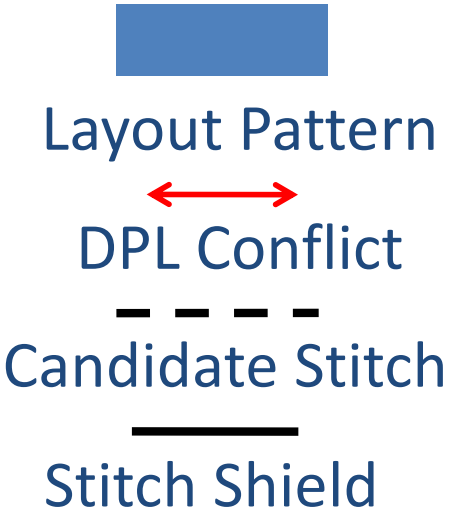
A Complete Example



Minimum Weighted Perfect Matching

Post-Processing for Dependent Stitches





- Two stitches that cannot co-exist may be produced by the algorithm
- If such situation occurs, re-run by disallowing each stitch in turn

Experiment Setup and Result

- Coded in C
- Simulation is run on a 2.8GHz Intel Linux machine with 32GB RAM
- Compare #stitches, #edge removals with [Xu+ ICCAD08]

	Face Merging Decomposer			[Xu+ ICCAD09]		
Design	# Stitch	# ER	CPU (s)	# Stitches	# ER	CPU (s)
AES	30	0	4.8	30	0	5.9
TOP-B	10892	637	42.6	10265	865	15.4
TOP-C	67581	3502	205.5	64385	4273	310.7
TOP-D	25853	1328	97.3	24767	1724	85

Conclusions

- Planar Conflict graph
- An odd face pairing based DPL decomposition framework
- Polynomial time matching based optimal solution
- Less stitches and conflicts while using much less runtime

References

- [Xu+ ISPD10] A Matching Based Decomposer for Double Patterning Lithography
- [Xu+ ICCAD09] GREMA: Graph Reduction Based Efficient Mask Assignment for Double Patterning Technology