UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

A Polynomial Time Exact Algorithm for Self-Aligned Double Patterning Layout Decomposition

Zigang Xiao

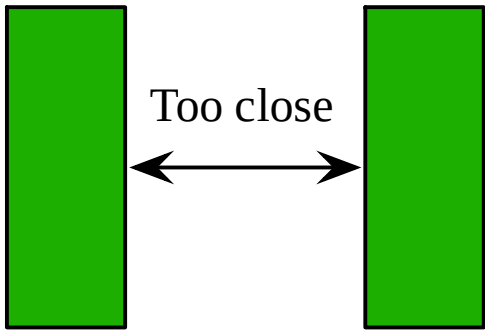
ISPD '12, Napa, CA
March 26, 2012

Outline

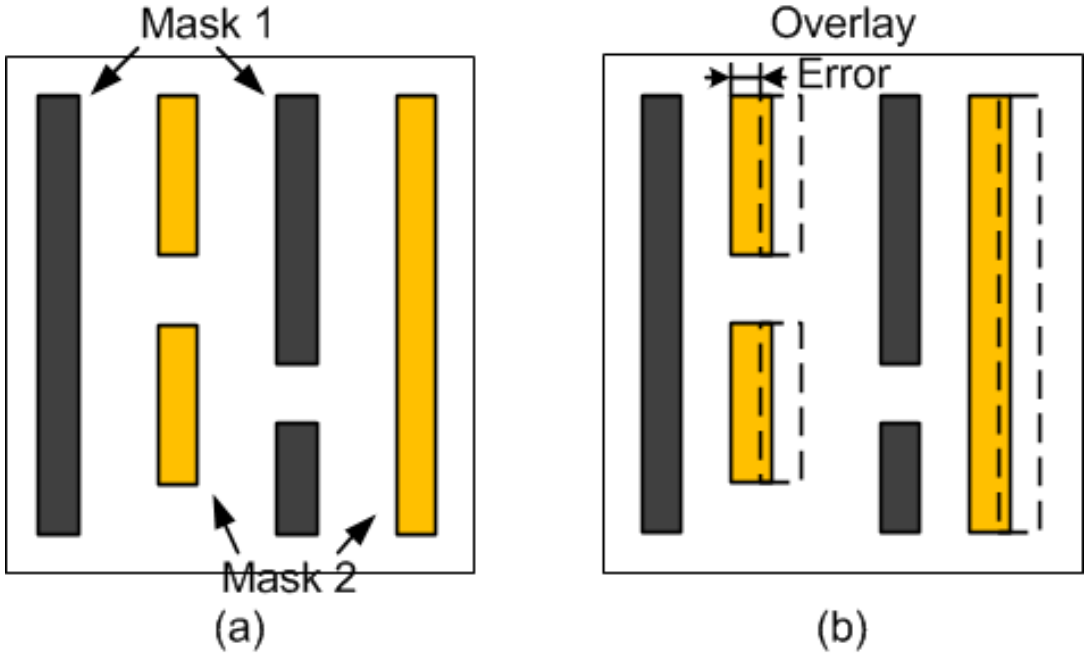
- Introduction
  - Background
  - SADP Process
  - Avoiding overlay in SADP
- Problem Formulation
- A Polynomial-time Exact Algorithm
- Experimental Results
- Summary

Background

- Single exposure lithography reaches its limit.
  - Double Patterning Lithography (DPL) is necessary.
- Problem of LELE DPL: overlay
  - Mask-to-mask misalignment
  - Ruin the process reliability, lead to yield degradation.
- Self-aligned Double Patterning (SADP) can avoid overlay.
  - Two mask phases: core mask, trim mask.



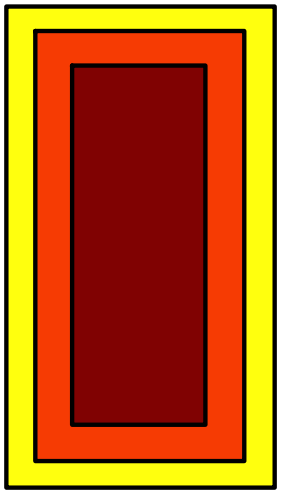
Cannot print in a single mask

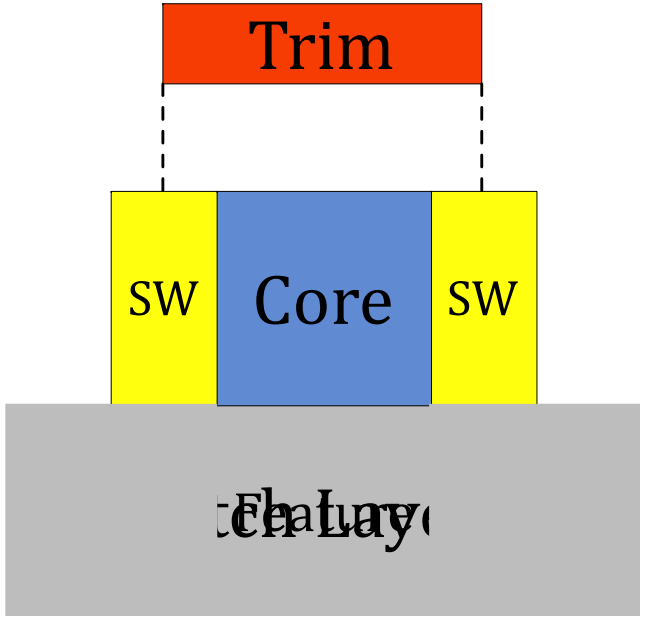


Overlay in traditional DPL

SADP Process

feature core sidewall trim

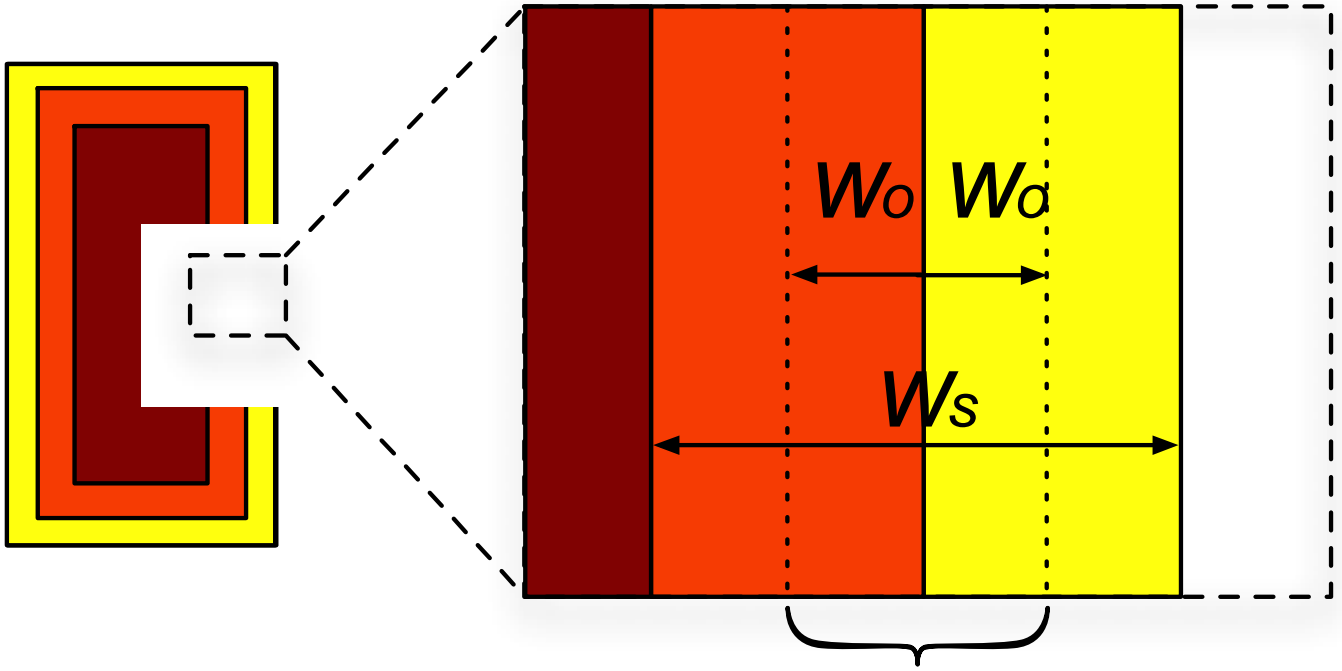




Patterns printed = trim & not sidewall

Crosscut View

Avoiding Overlay in SADP



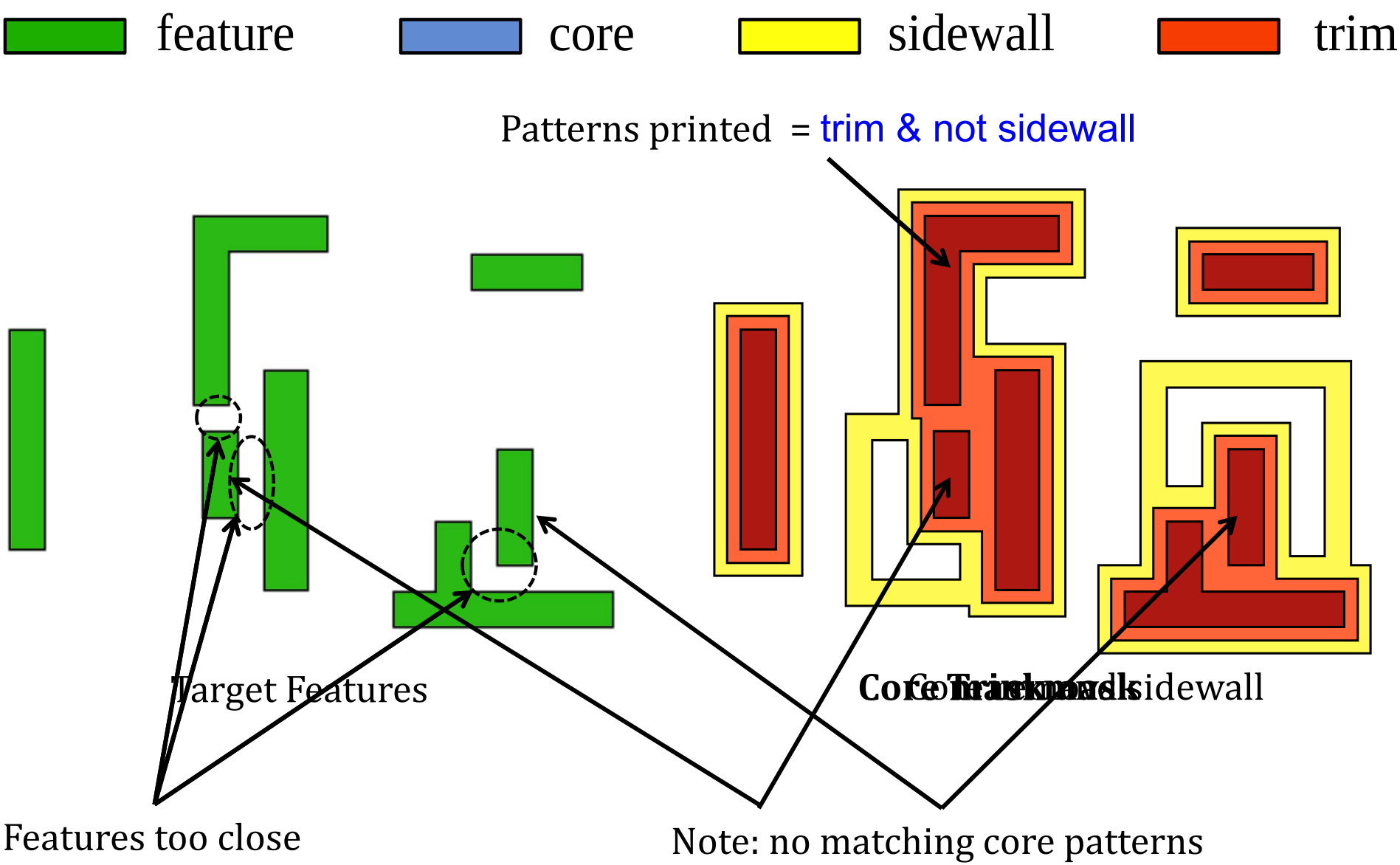
possible edge location of trim pattern after misalignment

W_o Maximum amount of trim mask misalignment

W_s Sidewall width

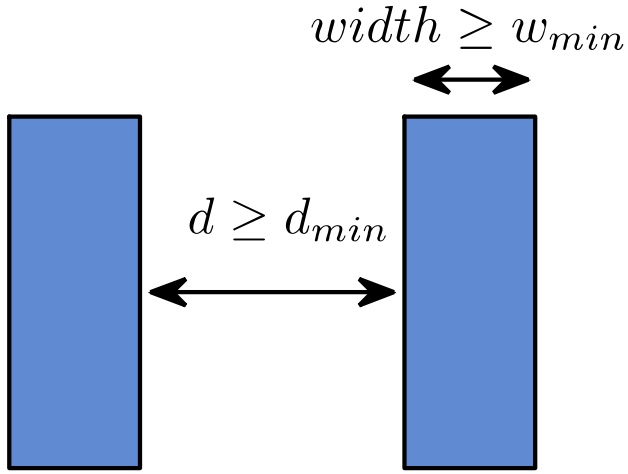
- Sidewall is providing protection for feature edges
- No overlay requirement: the boundaries of features must be **protected by sidewall**
  - Referred as **Ring of Sidewall**

SADP Process – Multiple Features



Problem Formulation

- Given a set of features, find a set of core patterns and trim patterns that can produce the features **exactly without overlay**.
  - Refer the solution as a decomposition.
- Process rules must not be violated
  - Minimum pattern distance d_{min}
  - Minimum pattern width w_{min}



Previous Works

- Minimization of overlay
- SAT [Zhang et al., SPIE Optical Lithography '11]
- ILP [Zhang et al. DAC '11]
  - Exact, but exponential time
- Graph approach [Ban et al. SPIE DFM through Design-Process Integration '11, DAC '11]
  - Not optimal

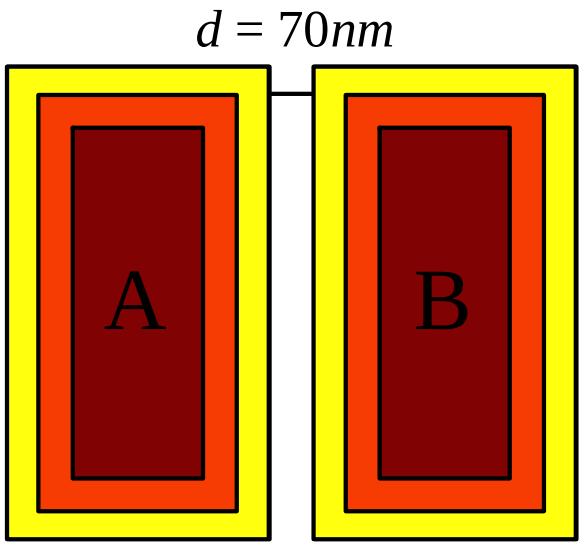
- Proposed algorithm
  - No-overlay solution
  - Polynomial time and exact

Outline

- Introduction
- Problem Formulation
- A Polynomial-time Exact Algorithm
  - Auxiliary core
  - Graph formulation
  - Details of the algorithm
- Experimental Results
- Summary

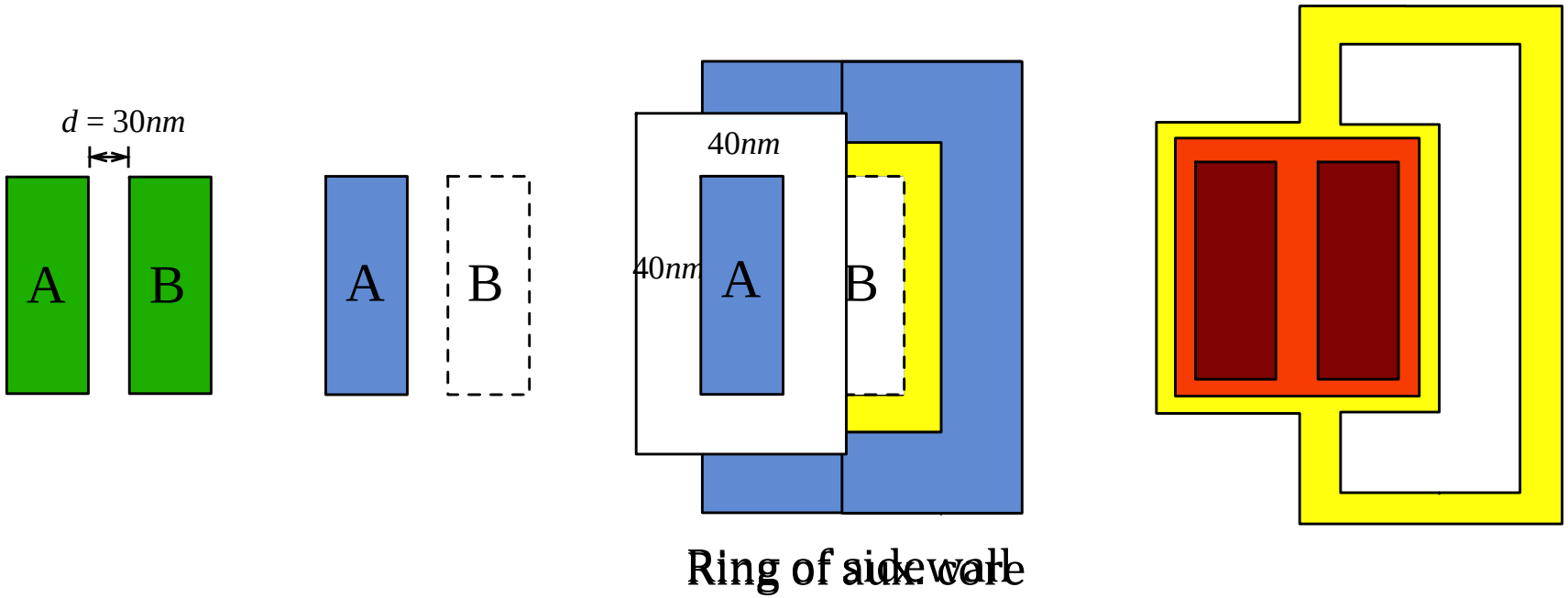
Auxiliary Cores

- Assume $d_{min} = w_{min} = 40nm$, $w_s = 30nm$, $w_o = 10nm$.
- *d:* distance between two features
- When $d < 30nm$, no solution (no sidewall possible).
- When $d \geq 40nm$, can use core patterns *simultaneously*:

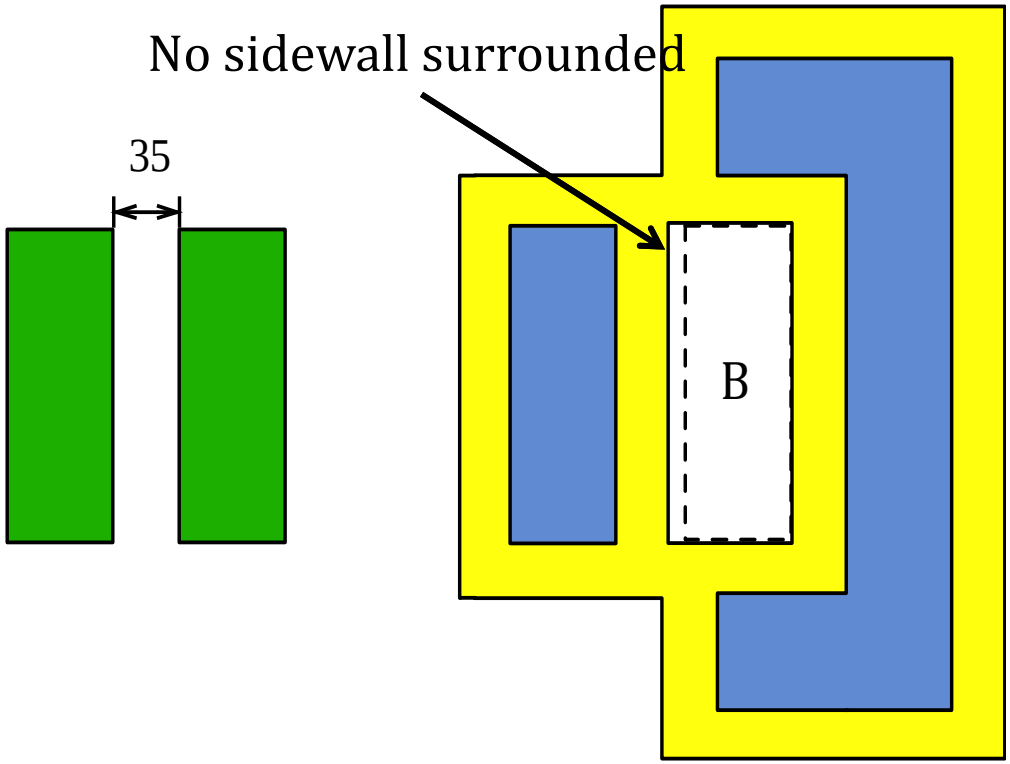


Auxiliary Cores – cont.

- When $d = 30nm$, cannot use two core patterns, use auxiliary cores to generate sidewalls.
  - Important way to *avoid overlay*!
- How about 30 < d < 40?



Feature Distance



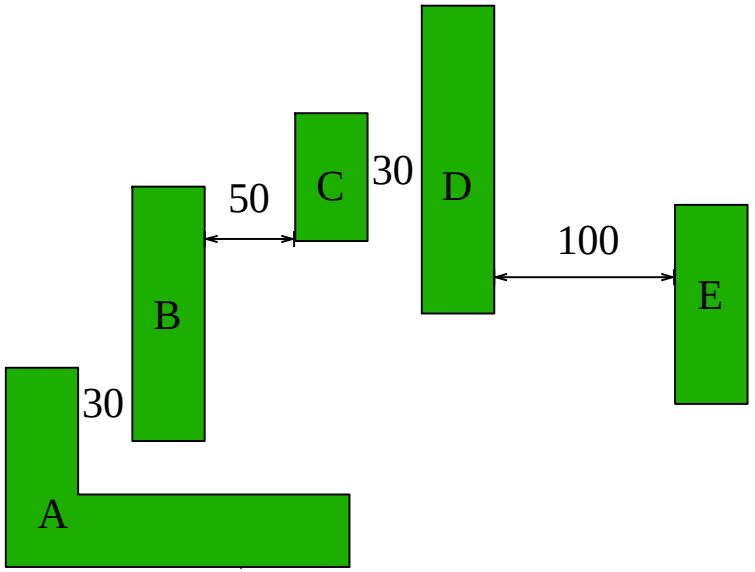
- Use two trim patterns: impossible (too close)
- Use one trim pattern: feature B is widened
- Cannot decompose when $30 < d < 40$

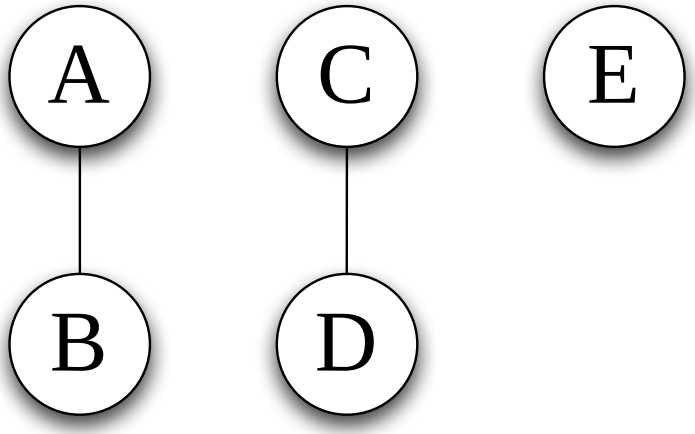
Summary

- Distance requirement for d
  - $d = 30$ (d_{min}) or $d >= 40$ (w_{min})
  - No solution if $w_s < d < d_{min}$

A Graph Formulation

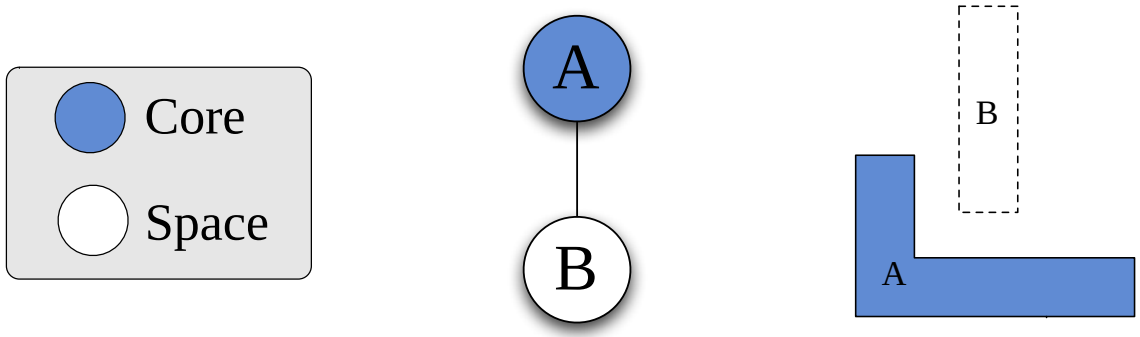
- We can reflect the distance constraints in a graph.
- Define SW-Graph *G=(V,E):*
  - *V*: include a vertex in *V* for every feature.
  - *E*: add an edge if *i* and *j* has distance $d=w_s$.
    - If $w_s < d < d_{min}$, **no solution**.





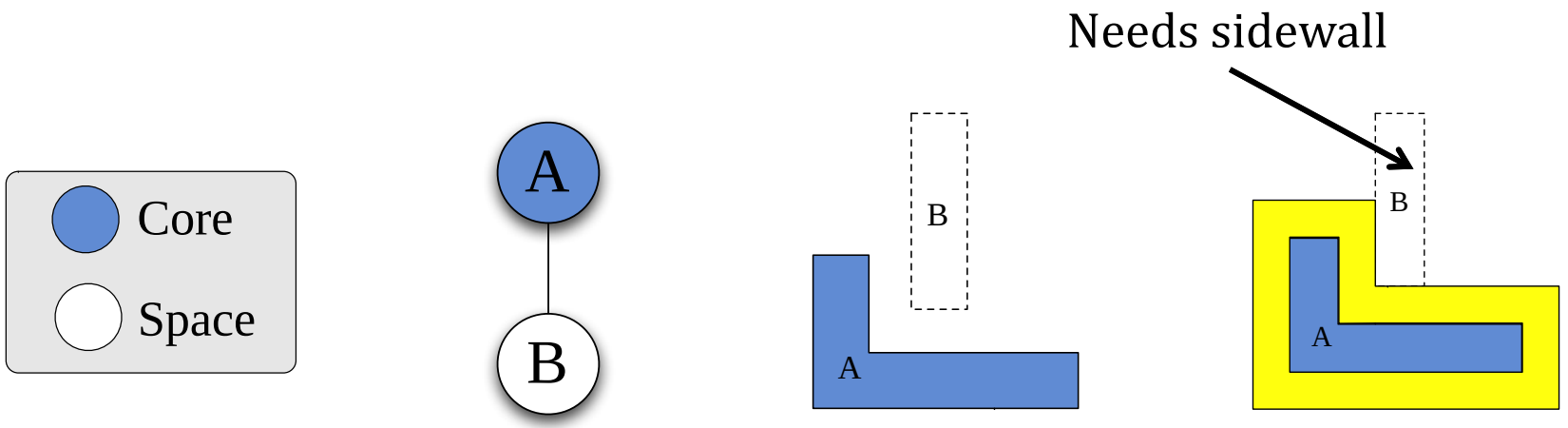
A Graph Formulation – cont.

- We can use **two-coloring** to find a set of core patterns that can be printed **simultaneously** in core mask.
  - Call the two colors as 'core' and 'space'.
  - Assign 'core' to feature A ↔ Put a core pattern at A
  - Assign 'space' to feature B ↔ leave B as blank



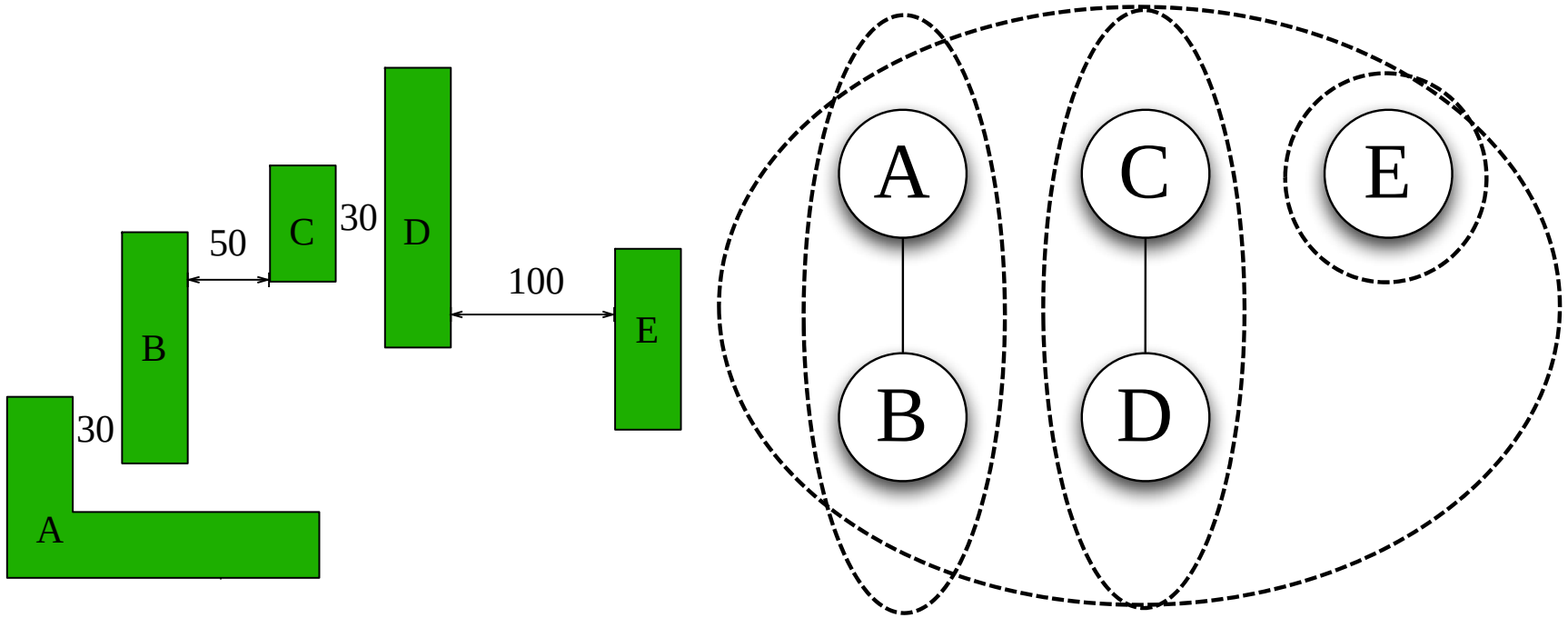
A Graph Formulation – cont.

- Two-colorability of SW-Graph is only **a necessary condition:** only features assigned 'core' have sidewalls, need a way to generate other required sidewalls.



- Note: LELE is solved by using two-coloring to assign colors (mask 1 or 2) to features
  - Two-coloring is not enough for our problem
  - The graph definitions are different

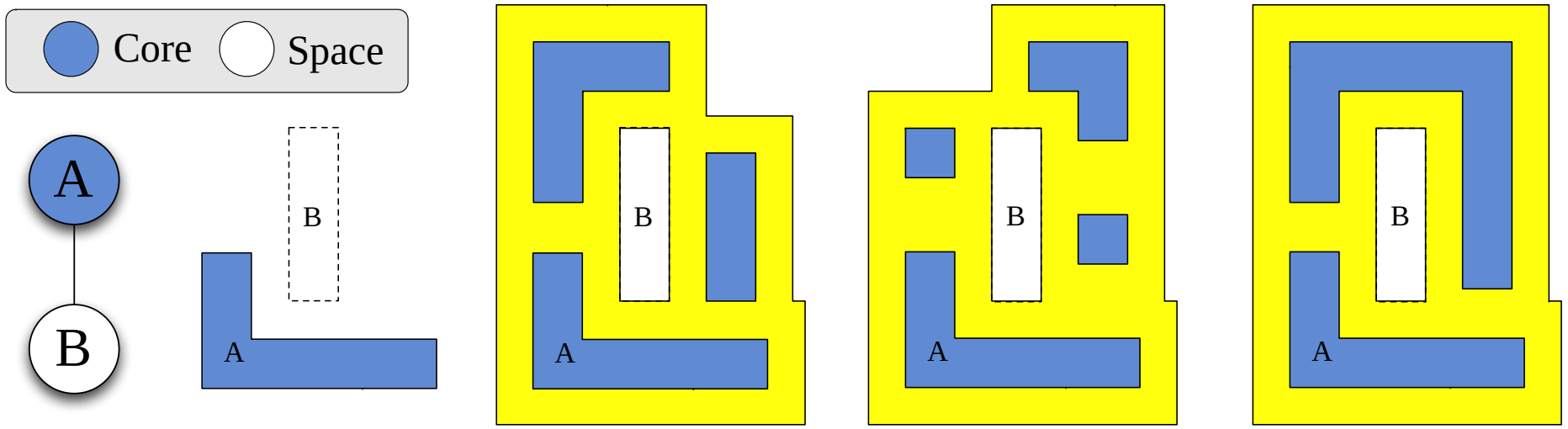
Overview



1. Find decompositions for each connected component in *G*.
2. Obtain a final decomposition from decompositions of connected components.

Step I: Decomposition for Component

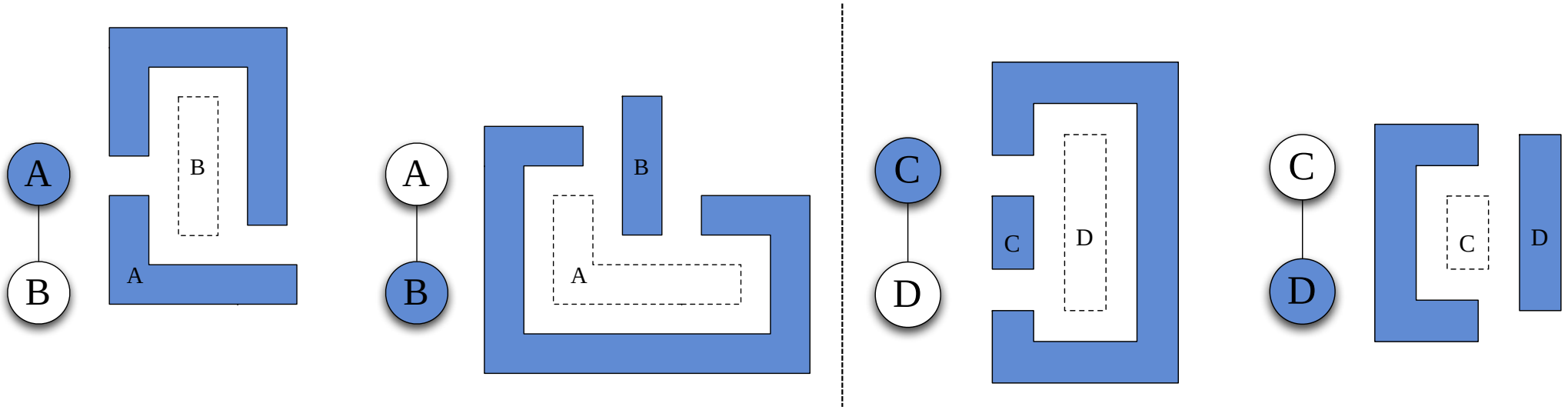
- Note: Given a two-coloring solution of a connected component, different decompositions may exist.



- Use ring of auxiliary core approach:
  - Remove conflict minimally
  - Merge when necessary
- Contains all possible auxiliary core settings.
- Referred as a *standard* decomposition.

Step I: Decomposition for Component

- Consequence
  - There are **two** two-coloring solutions for a colorable connected component.
  - A standard decomposition may not always exist.
- **At most** two standard decompositions for each component.



Decompositions for components {*A*, *B*}…

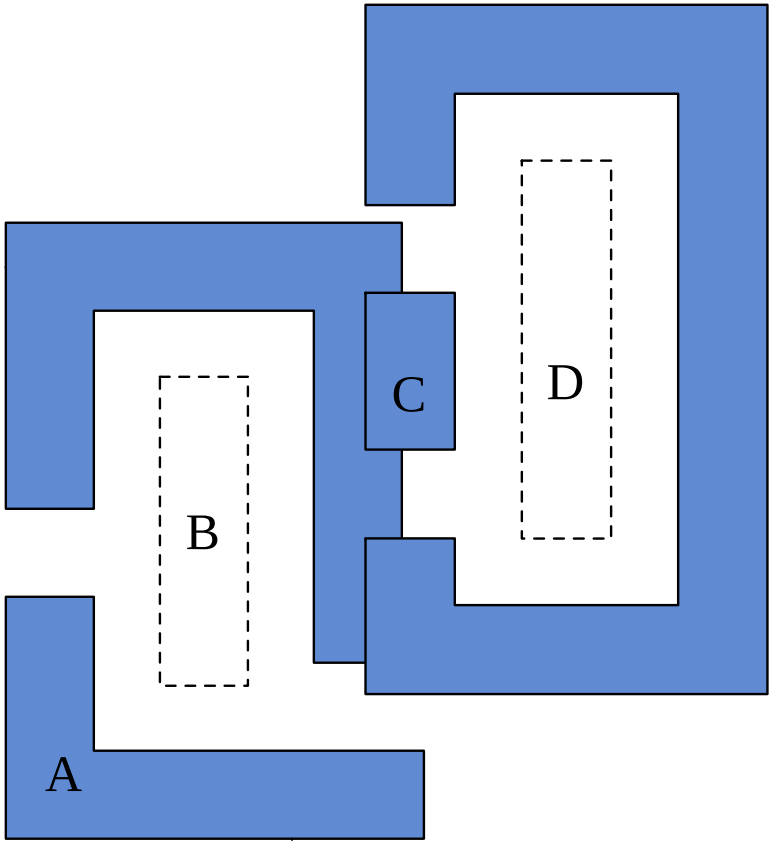
and {*C*, *D*}

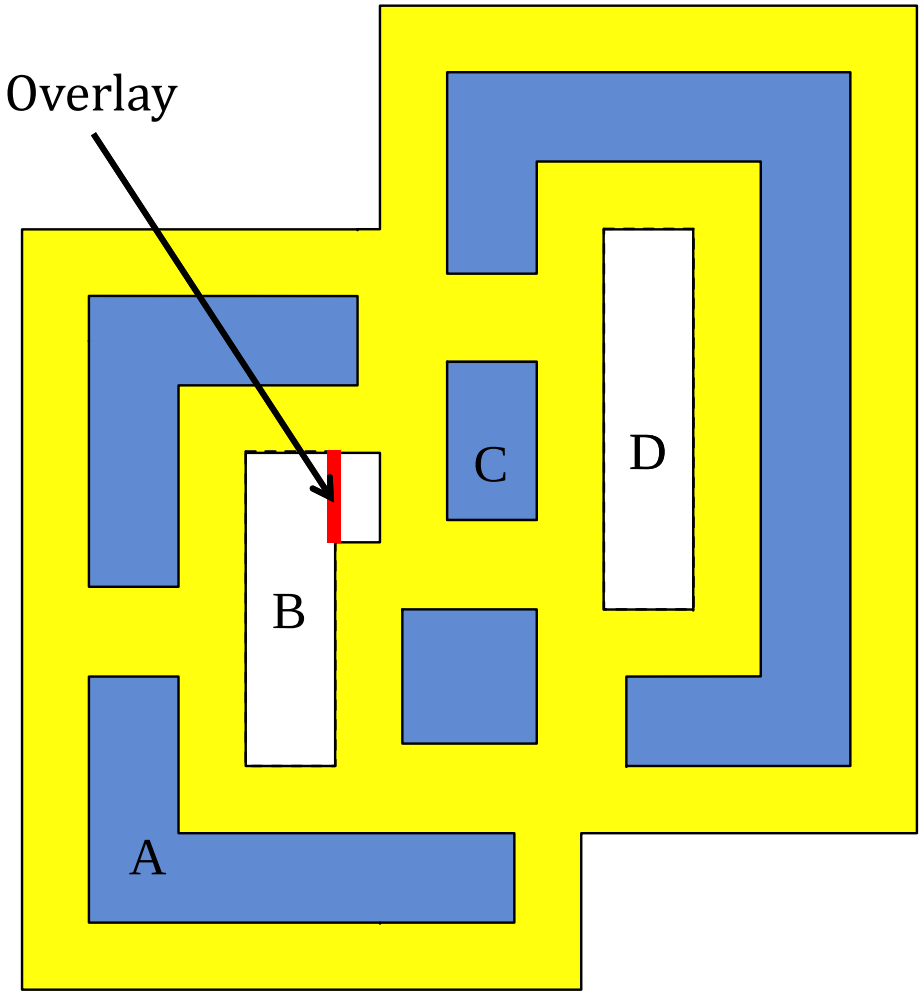
Step II: Combining Decompositions

- After step I, now we have decompositions for each connected component
  - $\{d_{11}, d_{12}\}, \{d_{21}, d_{22}\}, \ldots, \{d_{n1}, d_{n2}\}$
- Want to find a set of *compatible* decompositions from each connected component and combine them as a complete decomposition.
  - These decompositions must not introduce overlay.

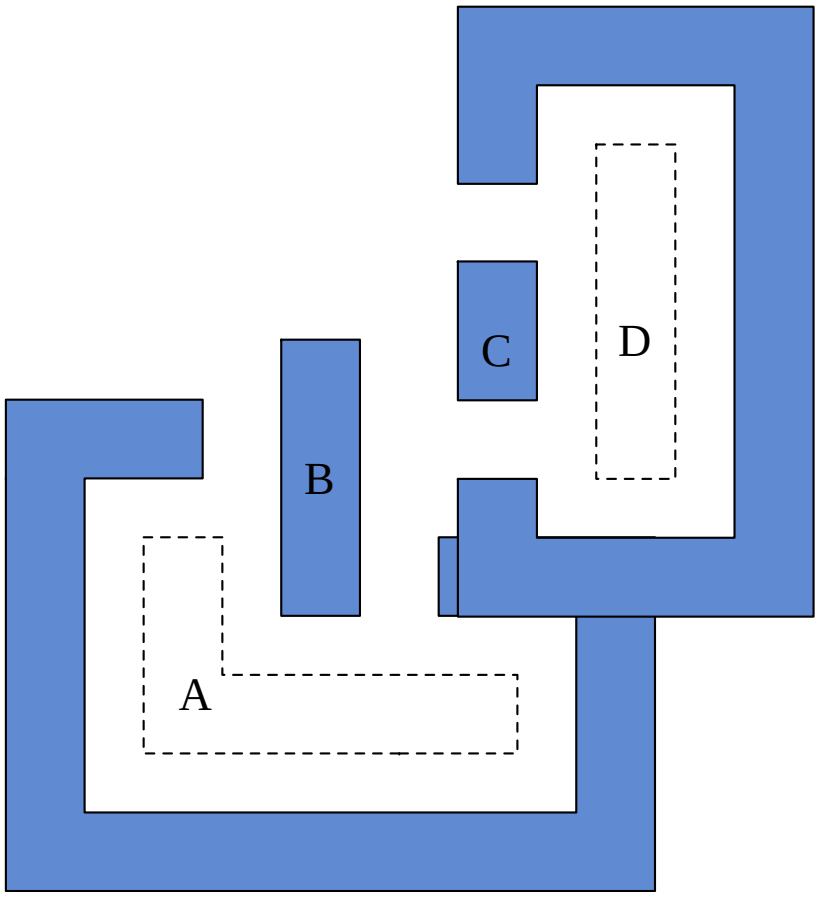
- Two decompositions are *compatible*, if combining them will **not** introduce overlay.
- Otherwise, they are *incompatible*.

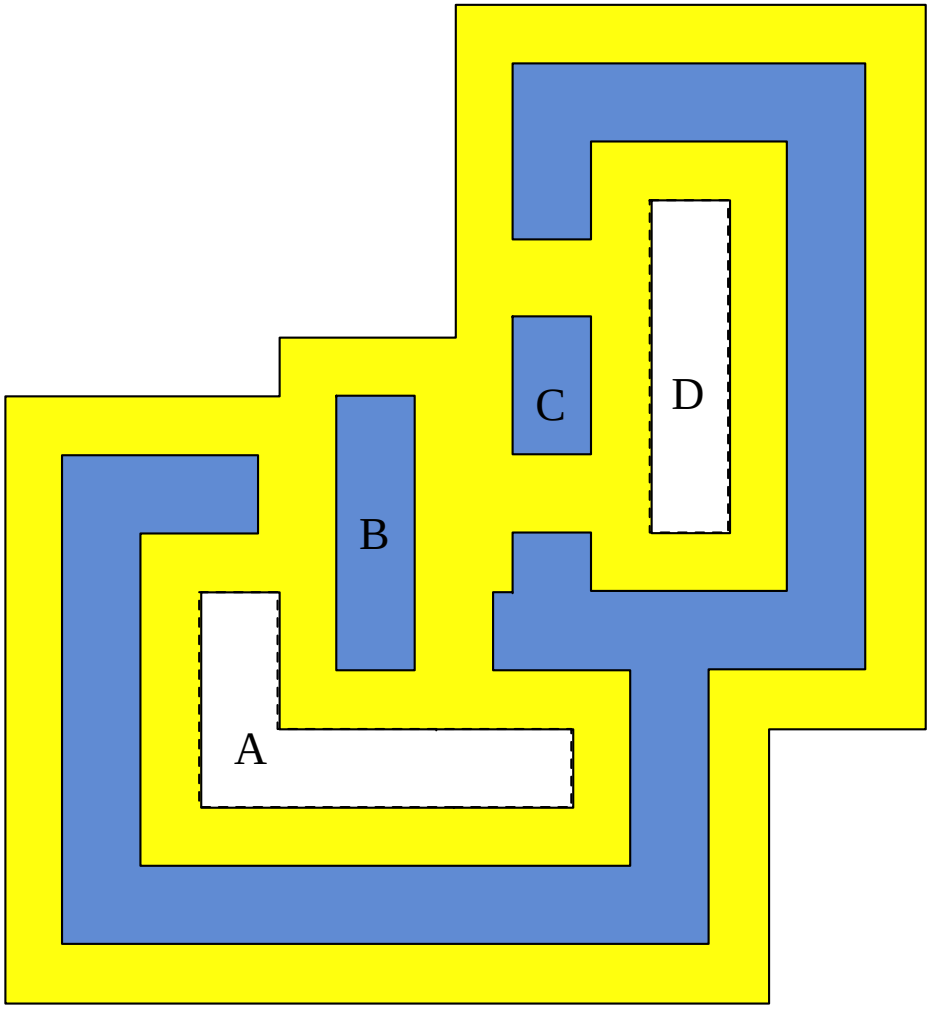
Example: Incompatible Decompositions





Example: Compatible Decompositions





All features are surrounded by sidewall - no overlay

Step II: Combining Decompositions - cont.

- Problem: find a decomposition d_i from each component i and combine them as a final decomposition.
  - d_i and d_j must be compatible for all $i \neq j$.

- There may be $O(n)$ components in G (n: # features).
- Each component have at most two decompositions.
- $O(2^n)$ combinations!
- How to find a set of compatible combinations efficiently?

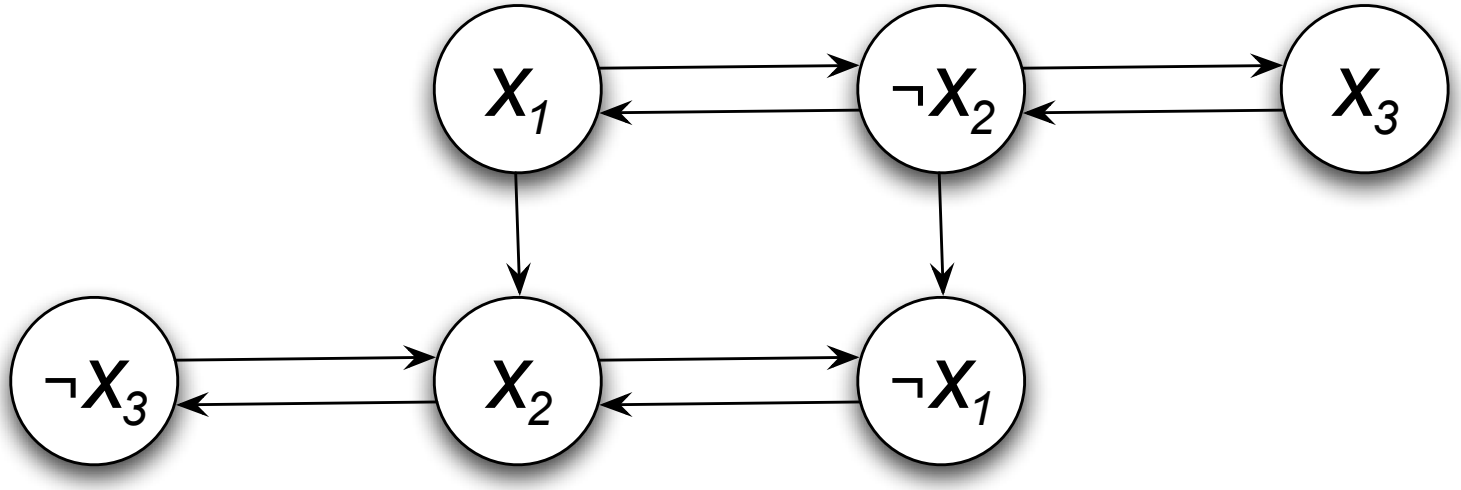
- It turns out this can be formulated as 2SAT problem and can thus solved efficiently.

Step II: Combining Decompositions - cont.

- Construct a Boolean **disjunction** f as follows:
  - Denote the two solutions of component i as x_i and $\neg x_i$. Either of them may *not* exist (no decomposition for coloring).
  - Let l be a literal (e.g., $\neg x_i$). If l does not exist, $f \leftarrow f \vee l$
  - If two solutions l_i and l_j from components i and j are not compatible, $f \leftarrow f \vee (l_i \wedge l_j)$. E.g., $x_i \wedge \neg x_j$.
  - At most $O(n^2)$ conjunctions, i.e., quadratic..
- Let $g=\neg f$, then a satisfiable assignment of g is a **final compatible decomposition**.
  - g = *true* $\leftrightarrow$ f = *false* $\leftrightarrow$ all terms in f are *false* $\leftrightarrow$ no conjunction is *true* $\leftrightarrow$ no incompatible decompositions appear together
- g is in 2CNF: 2SAT in 2CNF can be solved efficiently in linear time (Aspvall, Plass & Tarjan 1979).

Solution to the Example

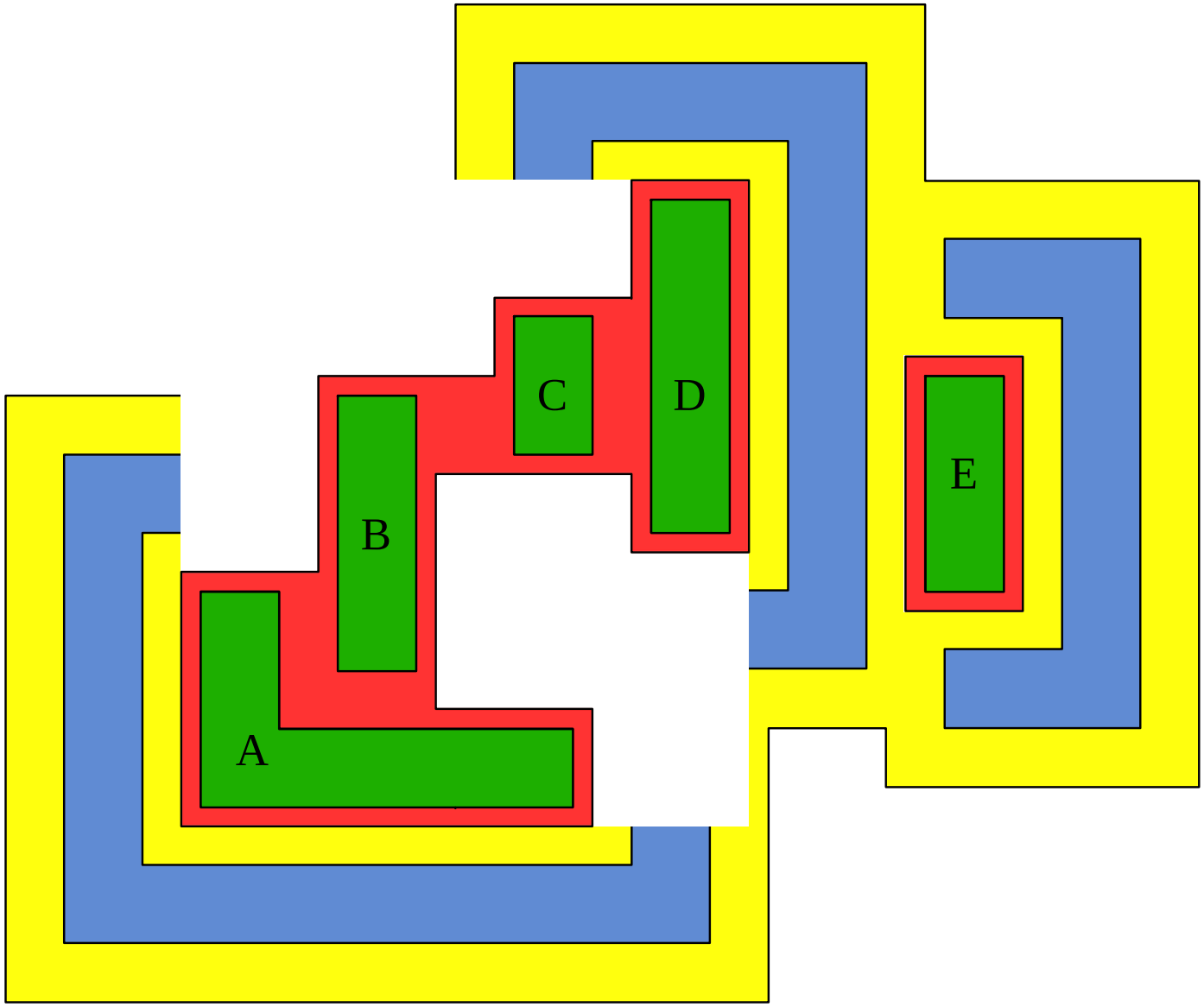
- Based on *implication graph* and its *strongly connected component* (Aspvall, Plass & Tarjan 1979)
- $f=(x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge \neg x_2) \vee (x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3)$
- $g=(\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3)$



Implication Graph of g

- A satisfiable assignment: $\neg x_1, x_2, \neg x_3$

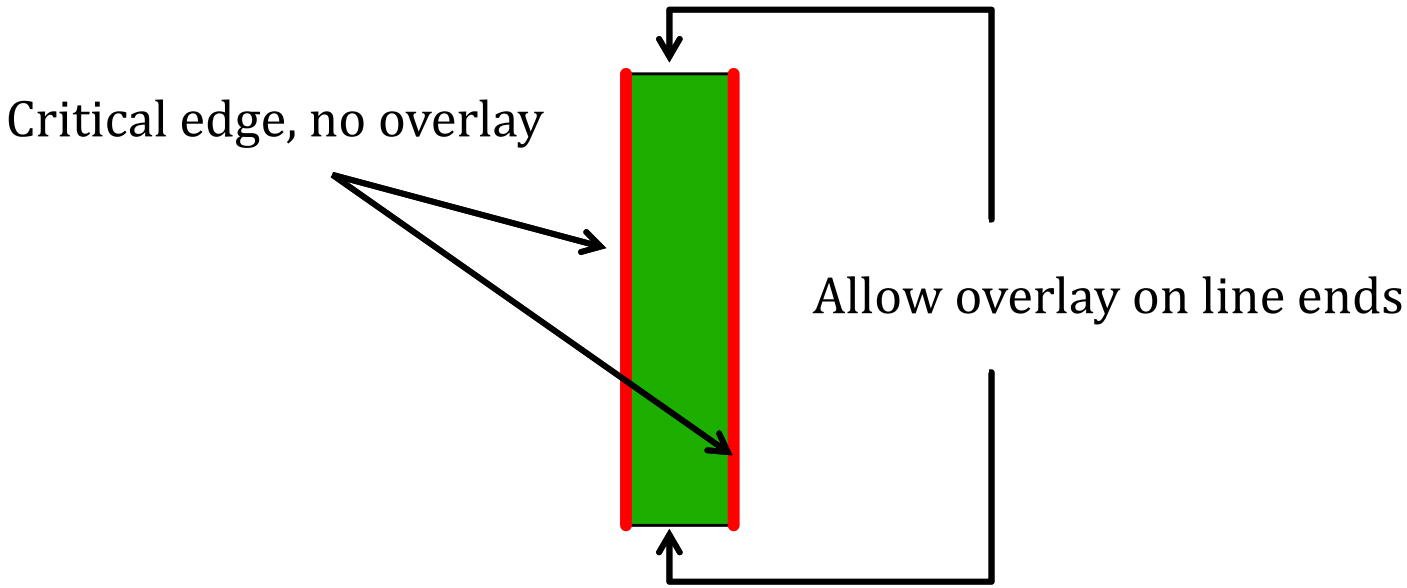
A Final Complete Decomposition



Core patterns obtained by combining $\neg x_1, x_2, \neg x_3$

Extension

- Allow overlay on non-critical edge



- Generalized, relaxed version
- Our algorithm can be modified to optimally solve this problem
  - Sidewall along critical edge

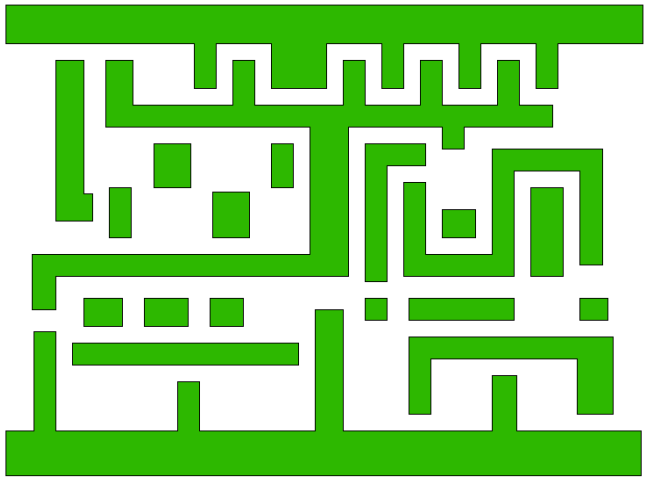
Experimental Results

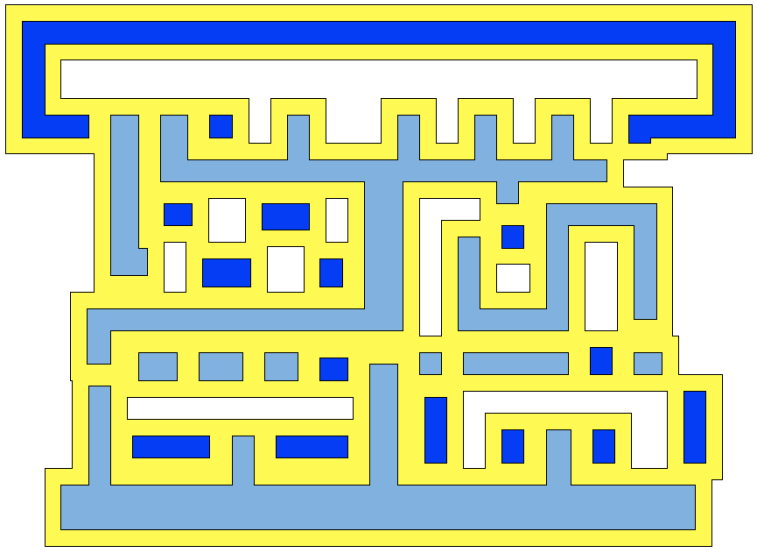
Name	# Features	# Components	CPU Time (s)	
			ours	ILP [1]
INV_X1	4	2	0.05	7.18
BUF_X1	5	1	0.03	3.45
BUF_X16	5	3	0.04	38.26
NAND2_X1	5	3	0.16	36.87
AND2_X1	6	2	0.07	25.33
OR2_X1	6	3	0.20	4.65
OR4_X4	8	2	0.18	5.22
Average			0.10	17.28

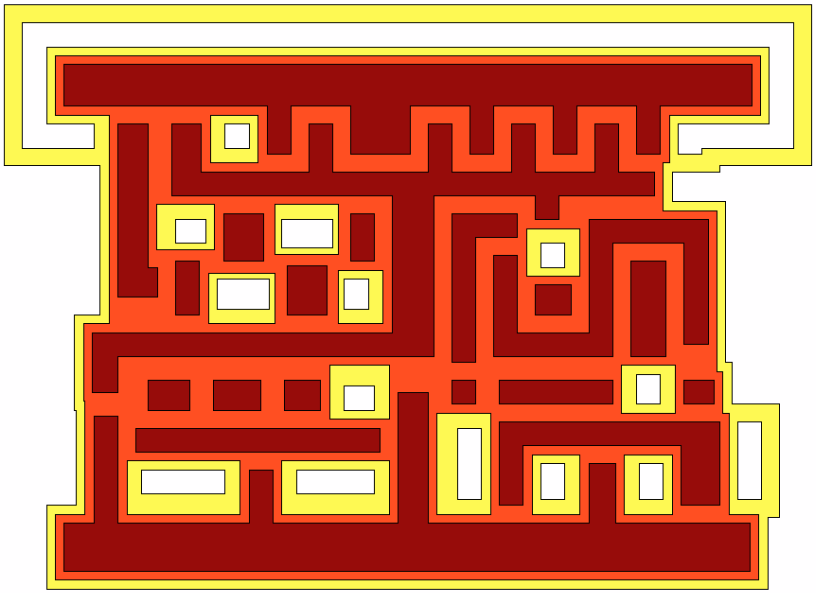
- Implemented in C++. Run on 3.20 GHz CPU w/ 32GB memory. Achieves ~100X Speed up.

[1] Zhang et al. Self-Aligned Double Patterning Decomposition for Overlay Minimization and Hot Spot Detection. DAC 2010

Example







Summary

- A polynomial-time exact algorithm for SADP decomposition on general 2D layout.
  - A correct graph formulation
  - Computes a decomposition without overlay efficiently.
- Experimental results demonstrate the efficiency of the proposed algorithm.

Thank you!