

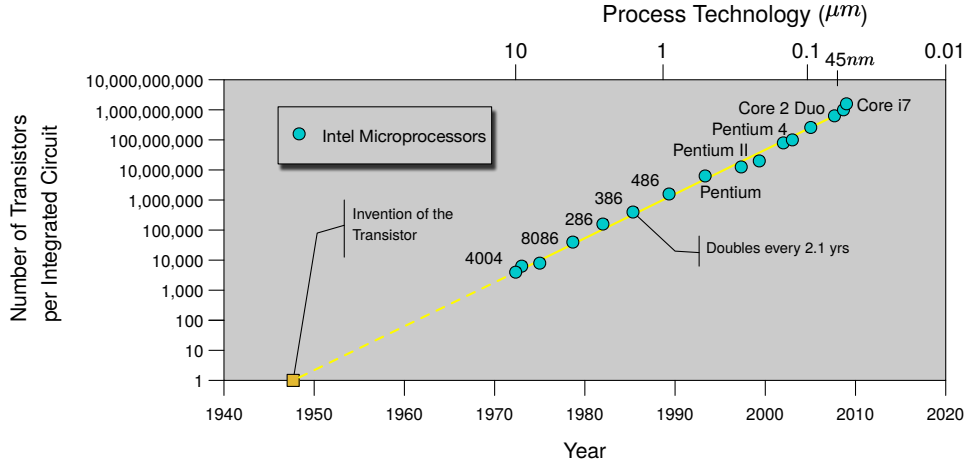
# OpenMPL: An Open Source Layout Decomposer

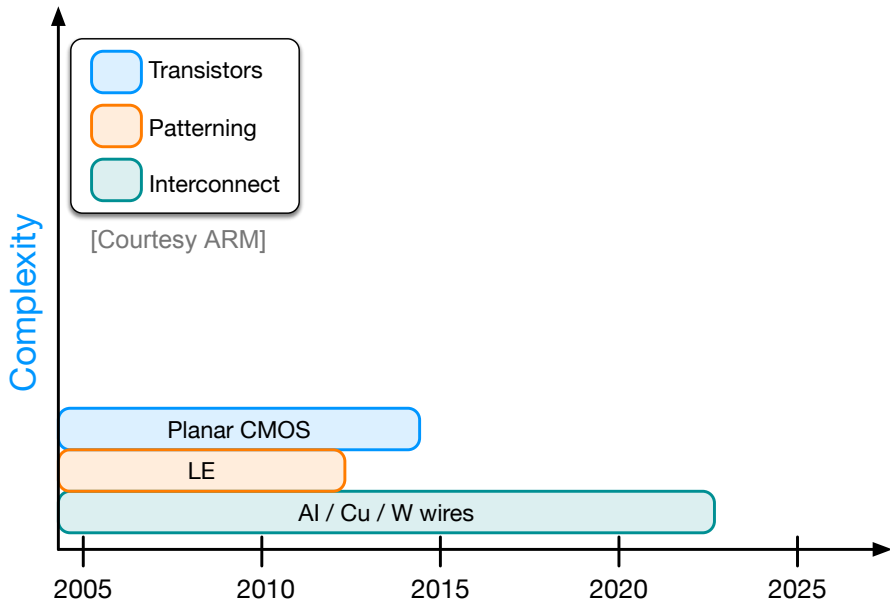
Wei Li<sup>1</sup>, Yuzhe Ma<sup>1</sup>, Qi Sun<sup>1</sup>, **Yibo Lin**<sup>2</sup>, Iris Hui-Ru Jiang<sup>3</sup>,  
Bei Yu<sup>1</sup>, David Z. Pan<sup>4</sup>

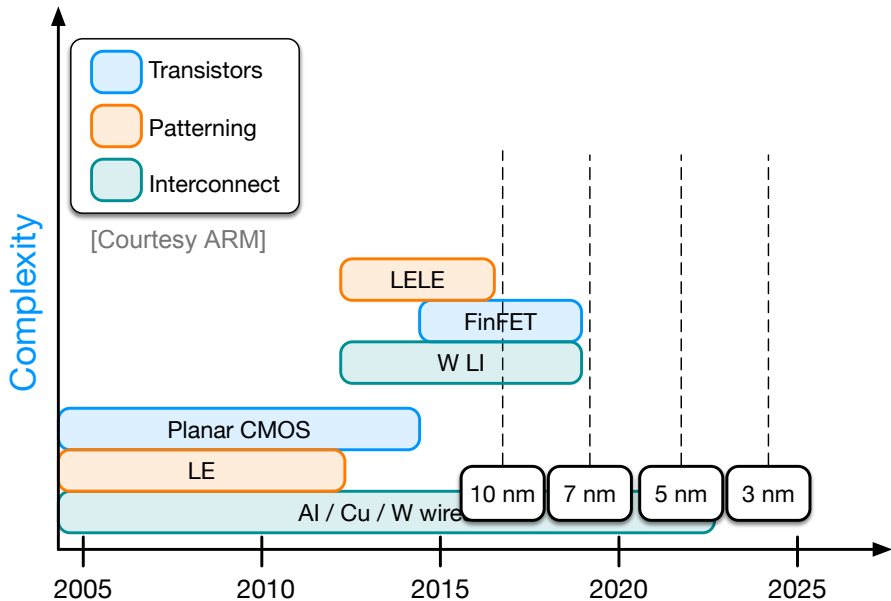
<sup>1</sup>The Chinese University of Hong Kong, <sup>2</sup>Peking University,  
<sup>3</sup>National Taiwan University, <sup>4</sup>University of Texas at Austin



# Moore's Law



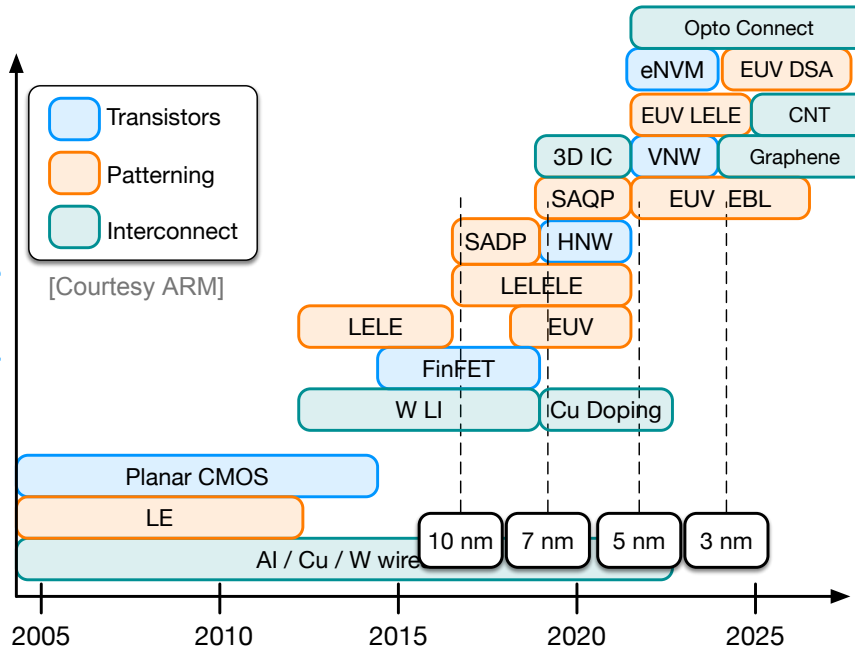




Complexity

- Transistors
- Patterning
- Interconnect

[Courtesy ARM]

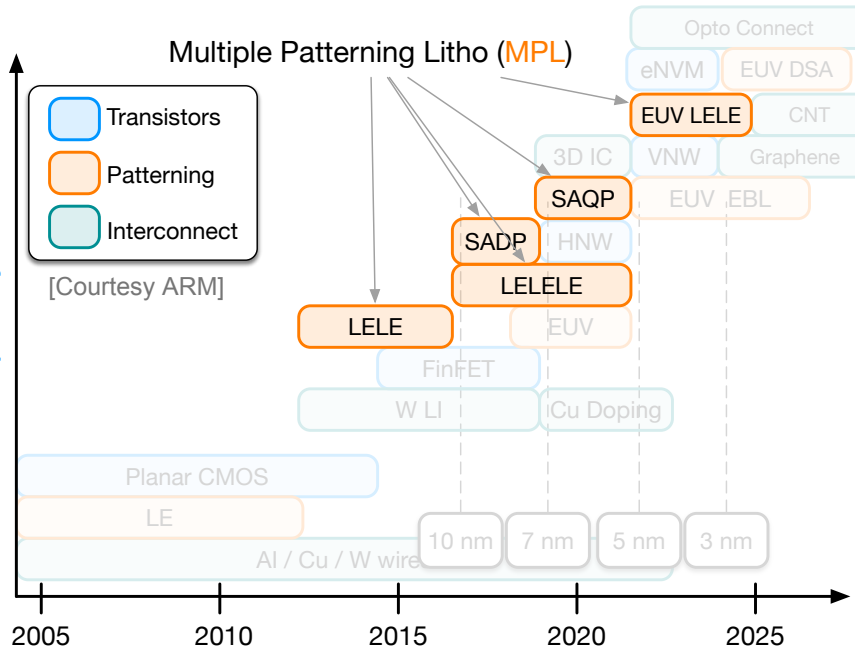


Complexity

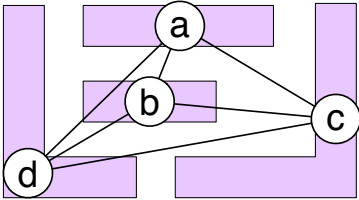
# Multiple Patterning Litho (MPL)

- Transistors
- Patterning
- Interconnect

[Courtesy ARM]



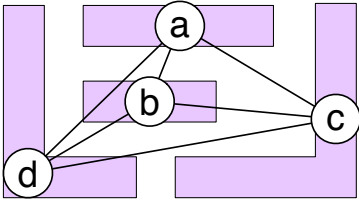
# Multiple Patterning Layout Decomposition



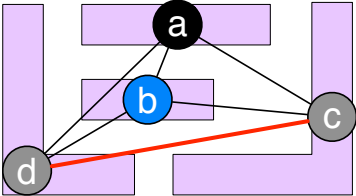
(a) Original layout



# Multiple Patterning Layout Decomposition



(a) Original layout

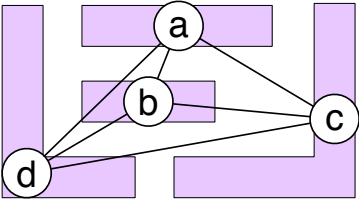


(b) TPL layout with conflicts

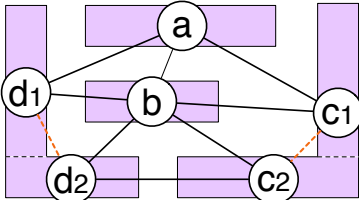




# Multiple Patterning Layout Decomposition



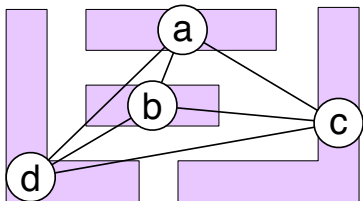
(a) Original layout



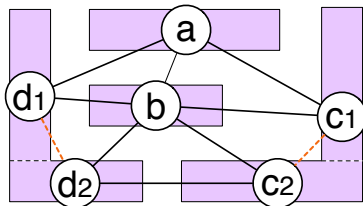
(b) Layout graph



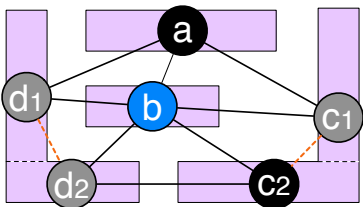
# Multiple Patterning Layout Decomposition



(a) Original layout



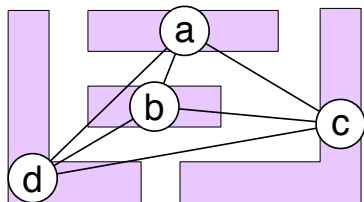
(b) Layout graph



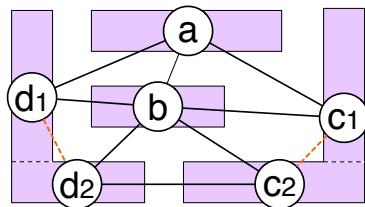
(c) Coloring on layout graph



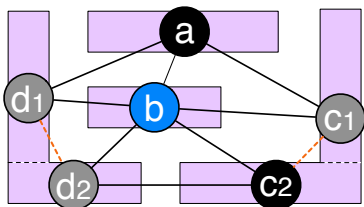
# Multiple Patterning Layout Decomposition



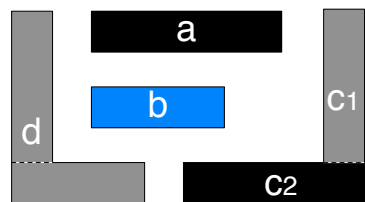
(a) Original layout



(b) Layout graph



(c) Coloring on layout graph

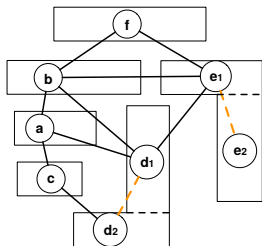


(d) Final decomposed layout



# Mathematical Formulation

- ▶ Some conflicts can be solved by **Stitch** Insertion.



## $k$ Pattern Layout Decomposition

$$\min_{\mathbf{x}} \sum_{e_{ij} \in CE} c_{ij} + \alpha \times \sum_{e_{ij} \in SE} s_{ij},$$

$$\text{s.t. } c_{ij} = \{x_i == x_j\},$$

$$s_{ij} = \{x_i \neq x_j\},$$

$$x_i \in \{0, 1, \dots, k\},$$

$$\forall e_{ij} \in CE,$$

$$\forall e_{ij} \in SE,$$

$$\forall i \in V.$$

- ▶  $x_i$  is a variable for the  $k$  available colors of the pattern  $v_i$ .
- ▶  $c_{ij}$  is a binary variable representing conflict edge  $e_{ij} \in CE$ .
- ▶  $s_{ij}$  stands for stitch edge  $e_{ij} \in SE$ .

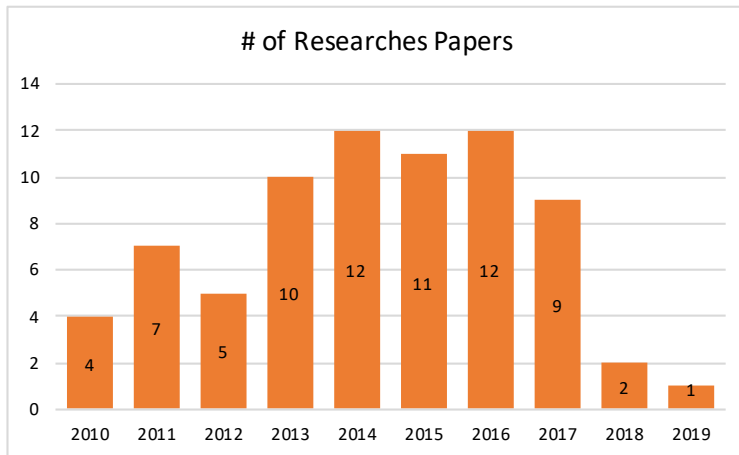


# Research Trend for Layout Decomposition

An **UNDERESTIMATION** from Google Scholar

Google Scholar

allintitle: "layout decomposition"



# Challenges in Layout Decomposition Research

## Complicated workflow and huge development overhead

- ▶ GDSII parsing and writing
- ▶ Layout graph construction and stitch insertion
- ▶ Graph simplification and core solvers



# Challenges in Layout Decomposition Research

## Complicated workflow and huge development overhead

- ▶ GDSII parsing and writing
- ▶ Layout graph construction and stitch insertion
- ▶ Graph simplification and core solvers

## Lack of reusable code and repeatability

- ▶ Need to develop from scratch
- ▶ Only share executable binaries instead of source code



# Challenges in Layout Decomposition Research

## Complicated workflow and huge development overhead

- ▶ GDSII parsing and writing
- ▶ Layout graph construction and stitch insertion
- ▶ Graph simplification and core solvers

## Lack of reusable code and repeatability

- ▶ Need to develop from scratch
- ▶ Only share executable binaries instead of source code

## No consistent framework for fair comparison

- ▶ Different implementations to the same algorithm
- ▶ Hard to make detailed comparison in depth





# OpenMPL 2.0: Open-Source Layout Decomposition Tool

limbo018 / OpenMPL

Unwatch 4 Star 8 Fork 6

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

An open multiple patterning framework Edit

Manage topics

312 commits 6 branches 7 releases 4 contributors BSD-3-Clause

Branch: master New pull request Create new file Upload files Find file Clone or download

limbo018 Merge branch 'develop' Latest commit 0eda4a7 3 minutes ago

bin	prepare for ASICON test	2 days ago
cmake	cmake and submodules in progress	3 months ago
images	add images to readme	3 days ago
erc	remove unused files	4 minutes ago
https://github.com/limbo018/OpenMPL		
.gitignore	stitch with only exactly one	2 months ago
.gitmodules	trial cmake	3 months ago
CMakeLists.txt	fix issue with boost timer and chrono order	3 days ago
LICENSE	Initial commit	last year
Readme.md	prepare for asicon release	4 hours ago
gdb.sh	OpenMPL	last year
run.sh	OpenMPL	last year

Readme.md

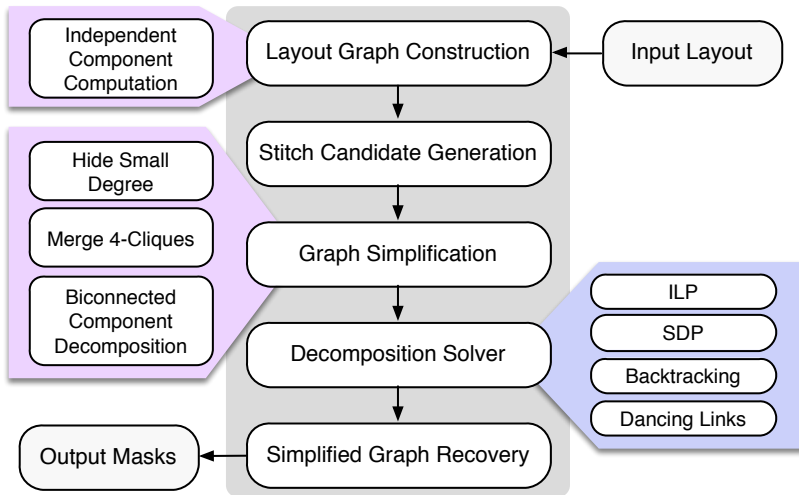
## OpenMPL

OpenMPL stands for open multiple patterning lithography framework.

Stitch Insertion	Graph Simplification	Decomposition
------------------	----------------------	---------------

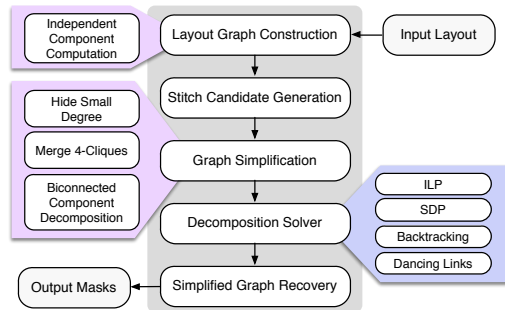


# Workflow



# Design Principles

- ▶ **Decoupled design stages**
- ▶ **Graphs for communications among kernel stages**
- ▶ **Efficiency and generality for different mask data**



# Extensible API to Incorporate New Algorithms

## ► Create a solver

```
// A solver class
class InstantColoring
    : public Coloring
{
public:
    // g is a boost::graph
    InstantColoring(g);
protected:
    virtual double coloring()
    {
        // override base function
        // return cost
    }
};
```



# Extensible API to Incorporate New Algorithms

## ▶ Create a solver

```
// A solver class
class InstantColoring
    : public Coloring
{
public:
    // g is a boost::graph
    InstantColoring(g);
protected:
    virtual double coloring()
    {
        // override base function
        // return cost
    }
};
```

## ▶ Integrate the solver

```
// define a solver
switch (algorithm)
{
    case BruteForce:
        solver = new BruteForce (g);
    case InstantColoring:
        solver = new InstantColoring (g);
    ...
}
...
// solve coloring
(*solver) ();
```



# OpenMPL: Open-Source Layout Decomposition Tool

We implement several layout decomposition solvers.

- ▶ **Backtracking**: Use DFS algorithm to find the solutions [Yu+,ICCAD'13].



# OpenMPL: Open-Source Layout Decomposition Tool

We implement several layout decomposition solvers.

- ▶ **Backtracking**: Use DFS algorithm to find the solutions [Yu+,ICCAD'13].
- ▶ **Integer Linear Programming**: Convert the problem into linear programming by binary encoding of vertex colors [Yu+,ICCAD'11].
- ▶ **Dancing Links**: Treat the problem as an exact cover problem and solve it in the BFS style [Chang+,DAC'16].



# OpenMPL: Open-Source Layout Decomposition Tool

We implement several layout decomposition solvers.

- ▶ **Backtracking**: Use DFS algorithm to find the solutions [Yu+,ICCAD'13].
- ▶ **Integer Linear Programming**: Convert the problem into linear programming by binary encoding of vertex colors [Yu+,ICCAD'11].
- ▶ **Dancing Links**: Treat the problem as an exact cover problem and solve it in the BFS style [Chang+,DAC'16].
- ▶ **Semidefinite Programming**: Use SDP to relax the problem and solve it in polynomial time [Yu+,ICCAD'11].





# Layout Simplification

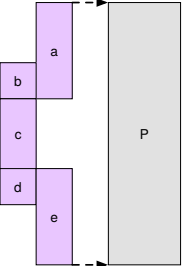
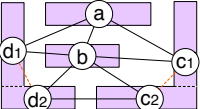
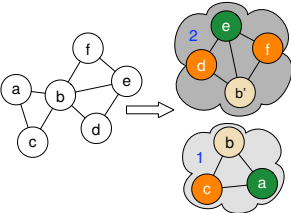
Reduce the graph size and therefore reduce the computation complexity.

- ▶ Level 0: **No Simplification**. Do not conduct any simplification.
- ▶ Level 1: **Hide small degree**. Temporarily remove nodes with low degree [Yu+,TCAD'15] [Lin+,SPIE'16].
- ▶ Level 2: **Merge 4-Clique**. Detect and merge sub-4-clique structures [Lin+,SPIE'16].
- ▶ Level 3: **Biconnected component decomposition**. Partition original graph into independent components [Kahng+,ICCAD'08].



# Additional Features

- ▶ **Multithreading:** Solve components in parallel.
- ▶ **Stitch Insertion:** Insert stitches to solve some conflicts.
- ▶ **Shape Friendly:** Specify the shape (POLYGON or RECTANGLE) easily.



# Command Line Options

Selected Options	Descriptions
-coloring_distance	a floating point number indicating number of coloring distance in nanometer
-color_num	an integer indicating number of masks (colors) < 3 4 >
-simplify_level	an integer indicating graph simplification level < 0 1 2 3 >
-path_layer	an integer indicating layer for conflict edges
-precolor_layer	an integer indicating layer for pre-colored patterns
-uncolor_layer	an integer indicating layer for coloring
-algo	algorithm type < ILP BACKTRACK LP SDP >
-shape	shape mode < RECTANGLE POLYGON >
-use_stitch	use stitch to avoid conflict
-gen_stitch	generate stitch candidate
-weight_stitch	weight of stitch
-thread_num	an integer indicating maximum thread number
-verbose	toggle controlling screen messages
-dbg_comp_id	debug component id



# Experimental Settings

- ▶ Triple Patterning Lithography
- ▶ Conducted on modified ISCAS benchmarks from [Yu+,ICCAD'13]
- ▶ Coloring distance:  $120nm$  for the first ten cases and  $100nm$  for the last five cases.



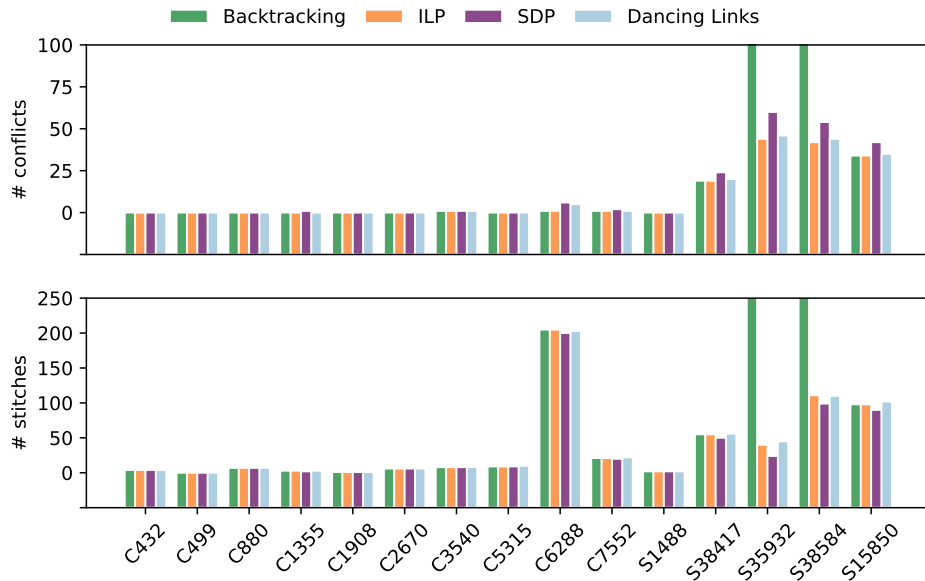
# Experimental Results

(a) total\_c1, TPLD, SDP coloring process.

(b) total\_c2, TPLD, SDP coloring process.



# Experimental Results – Quality



# Experimental Results – Runtime



# Quality and Runtime Tradeoffs

	Quality	Efficiency
Backtracking	★ ★ ★ ★ ★	★
ILP	★ ★ ★ ★ ★	★ ★
SDP	★ ★ ★	★ ★ ★ ★
Dancing Links	★ ★ ★ ★	★ ★ ★ ★ ★

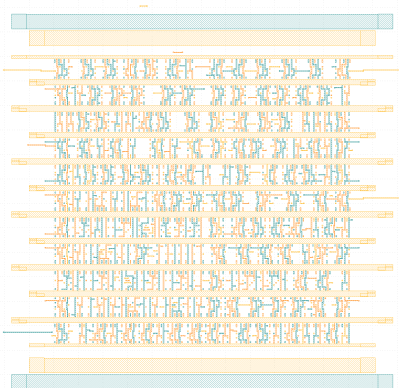




# Conclusion

## Open-source layout decomposition framework

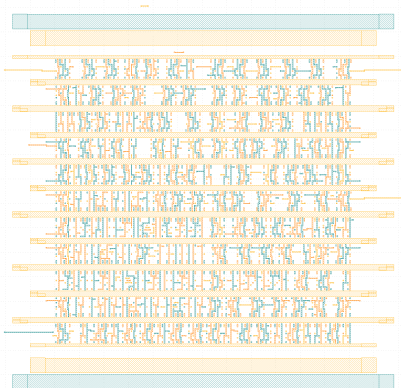
- ▶ Four layout decomposition solvers;
- ▶ Four graph simplification strategies;
- ▶ Multithreads, stitch insertion supported and shape frinedly.



# Conclusion

## Open-source layout decomposition framework

- ▶ Four layout decomposition solvers;
- ▶ Four graph simplification strategies;
- ▶ Multithreads, stitch insertion supported and shape frinedly.



## Future Work

- ▶ Efficiency: hardware acceleration.
- ▶ Quality: post refinement.



# Thank You



Wei Li ([wli@cse.cuhk.edu.hk](mailto:wli@cse.cuhk.edu.hk))

Yuzhe Ma ([yzma@cse.cuhk.edu.hk](mailto:yzma@cse.cuhk.edu.hk))

Qi Sun ([qsun@cse.cuhk.edu.hk](mailto:qsun@cse.cuhk.edu.hk))

Yibo Lin ([yibolin@pku.edu.cn](mailto:yibolin@pku.edu.cn))

Iris Hui-Ru Jiang ([huiru.jiang@gmail.com](mailto:huiru.jiang@gmail.com))

Bei Yu ([byu@cse.cuhk.edu.hk](mailto:byu@cse.cuhk.edu.hk))

David Z. Pan ([dpan@ece.utexas.edu](mailto:dpan@ece.utexas.edu))

