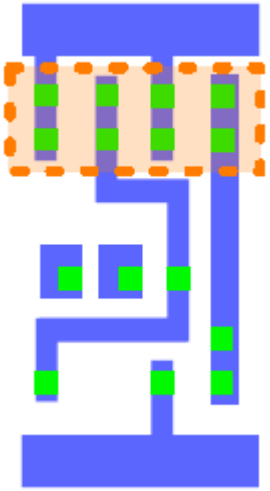
Triple Patterning Aware Detailed Placement

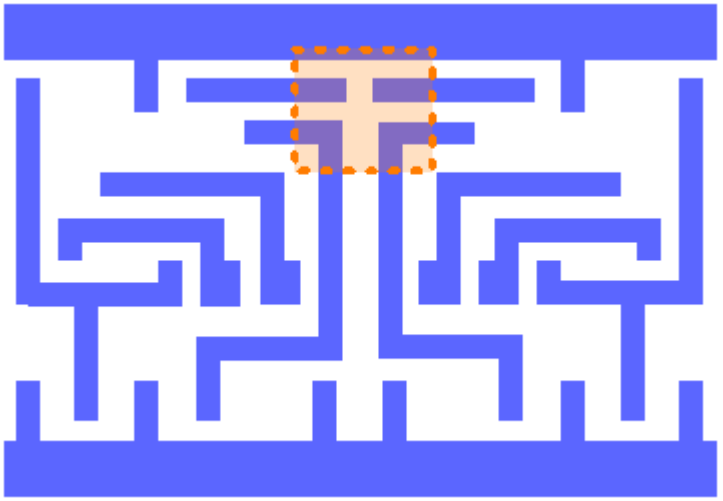
Methodology for Standard Cell Compliance and Detailed Placement for Triple Patterning Lithography

Bei Yu, Xiaoqing Xu, Jhih-Rong Gao, David Z. Pan

Native Conflicts for TPL

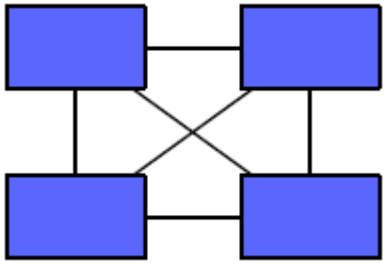


native conflict
within a cell

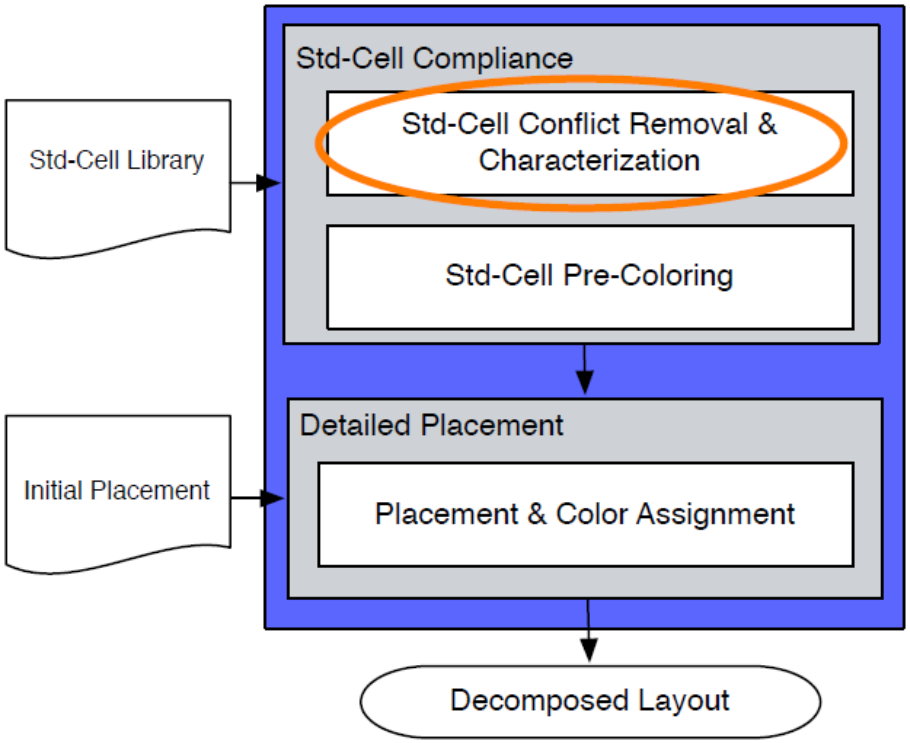


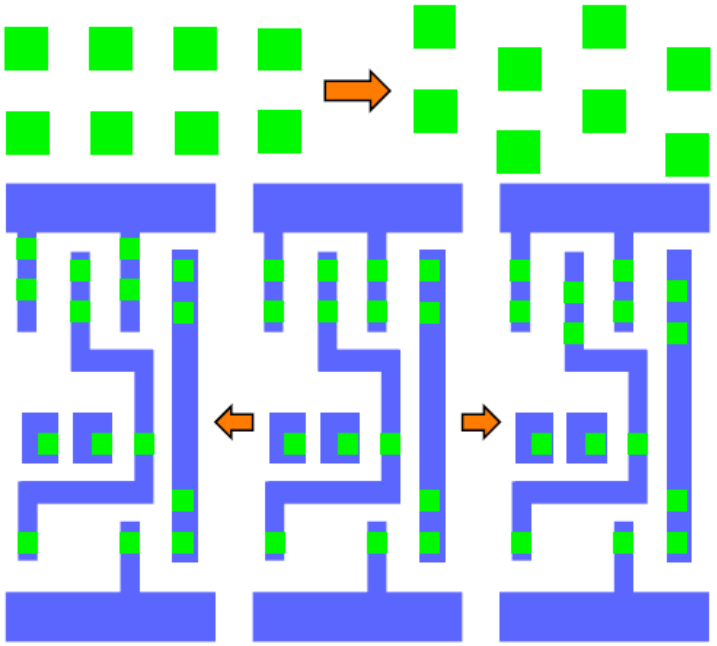
native conflict
between adjacent cells

- 4-cliques are not TPL decomposable



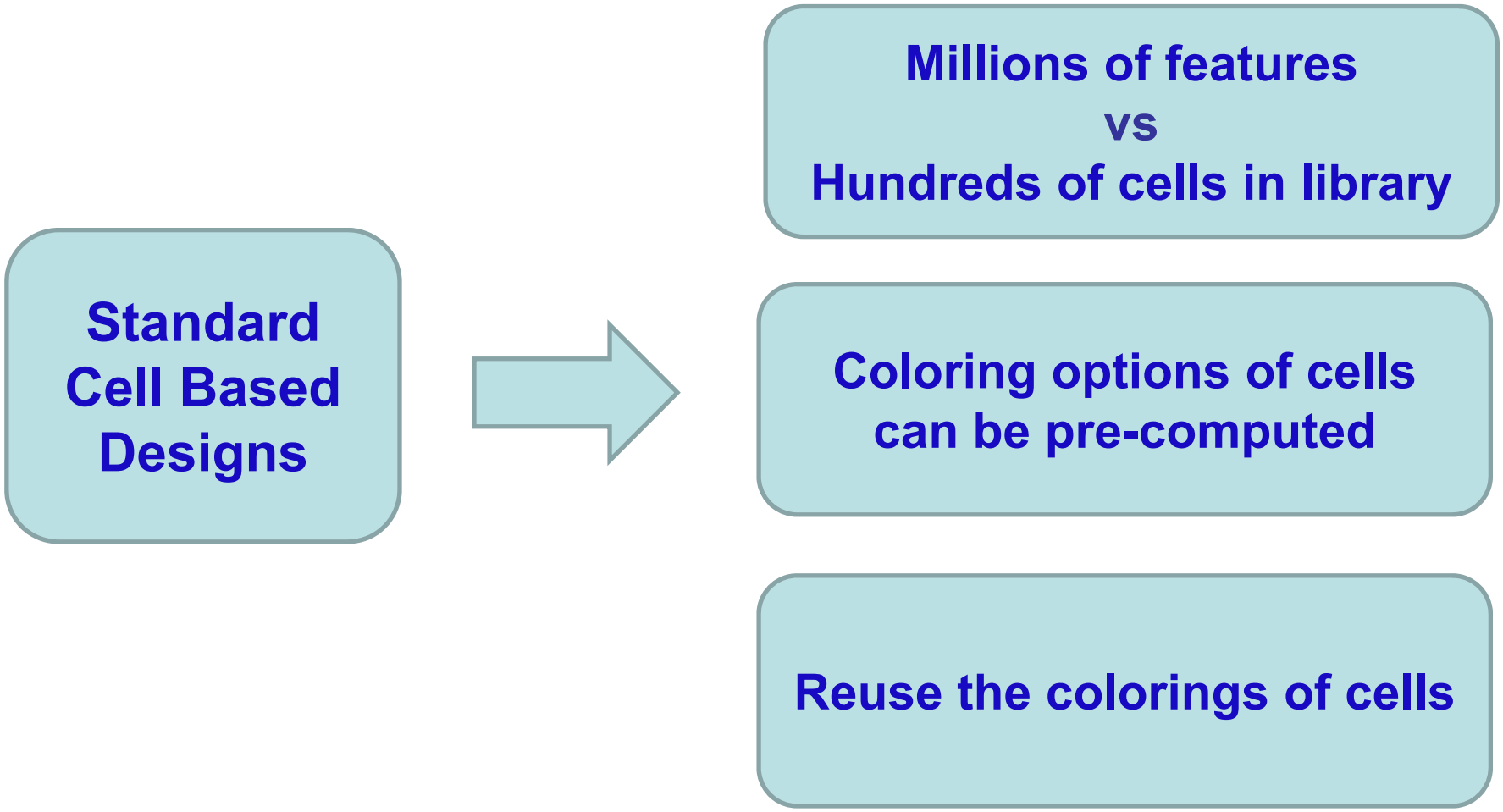
Standard Cell Native Conflict Removal





Removing inner cell native conflict of contacts by shifting

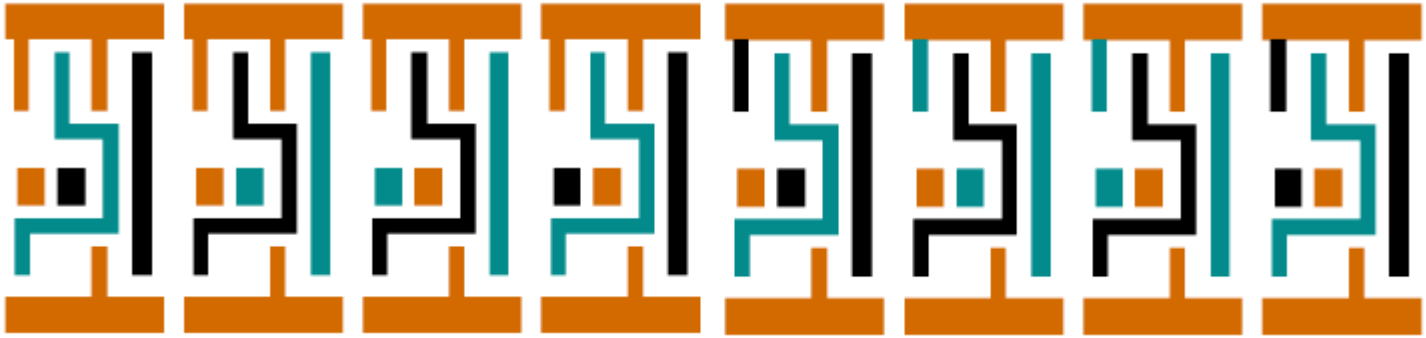
Hierarchical Approach



Standard Cell Pre-Coloring

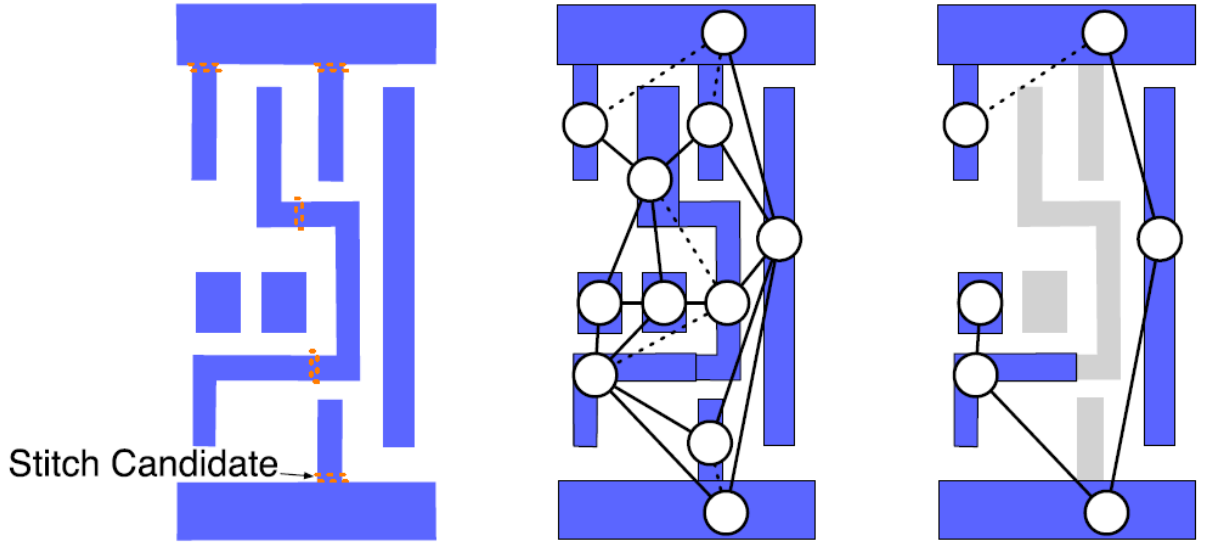
- We want to find all feasible colorings of a cell with a small stitch count (Why?)

e.g. Pre-coloring solutions of a cell with 0 or 1 stitch



Standard Cell Pre-Coloring

- A cell feature at distance $\geq d_{min}$ from cell boundaries will not conflict with any other cell



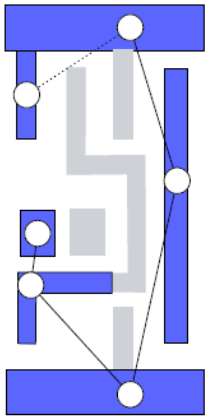
Cell layout with all stitch candidates

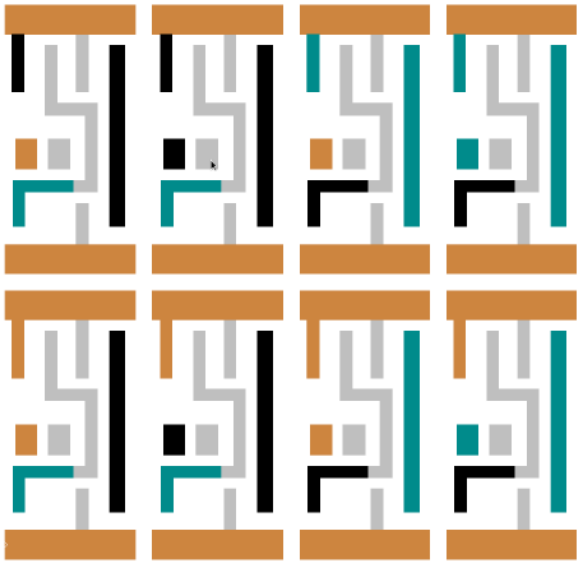
Conflict graph w/ conflict edges and stitch edges

Simplified conflict graph (SCG) after removing immune features

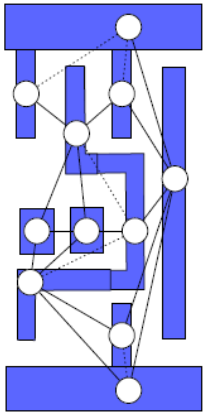
Standard Cell Pre-Coloring

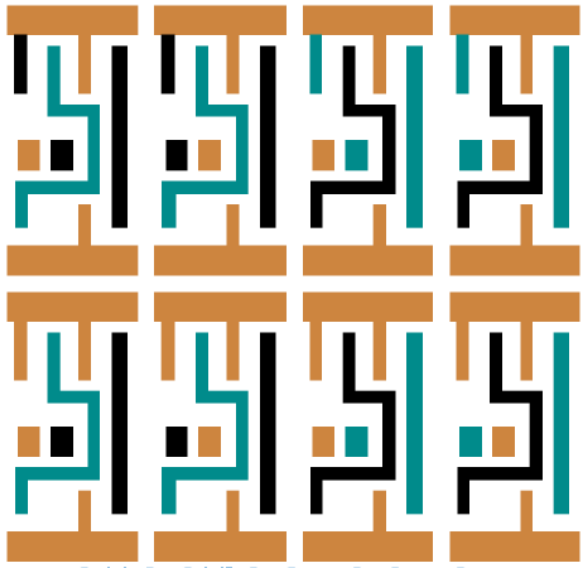
Stage 1: Enumerate all feasible coloring for SCG with power/ground rails assigned default color (may only keep those with small stitch counts)





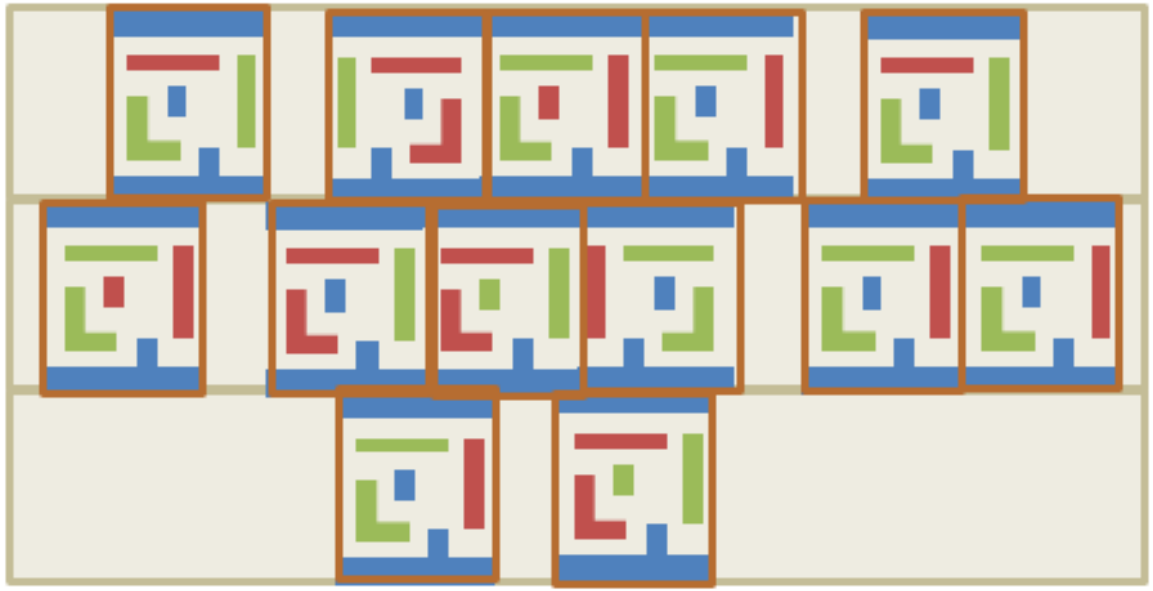
Stage 2: For each pre-colored SCG, check if immune features can be colored and keep non-redundant solutions with small stitch counts





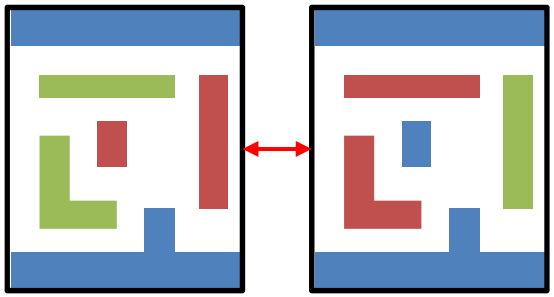
Standard Cell Placement

- All cells have the same height
- Power/Ground rails are at the top and bottom of cell
- Placed cells are aligned to placement sites
- No inter-row coloring conflicts due to wide P/G rails



Adjacent Standard Cell Separation

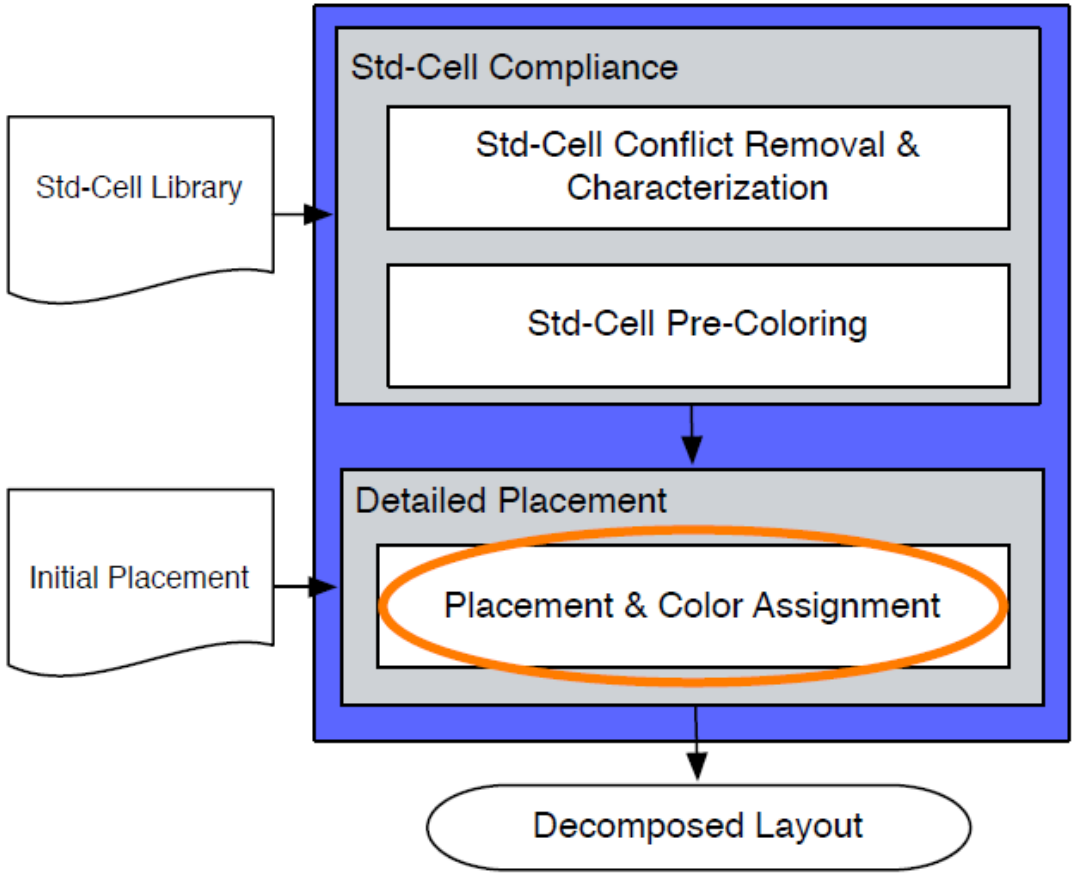
- Minimum separation between two adjacent standard cells on a row depends on their colorings



- Can be pre-computed and stored in a table
  - *dist(i,p,j,q)* : minimum feasible distance of cell *i* to cell *j* if *i* is on the left of *j* and cell *i* uses its *p*-th coloring option and cell *j* uses its *q*-th coloring option for all *i,p,j,q*

TPL Aware Detailed Placement

- Start with an optimized but TPL unaware detailed placement



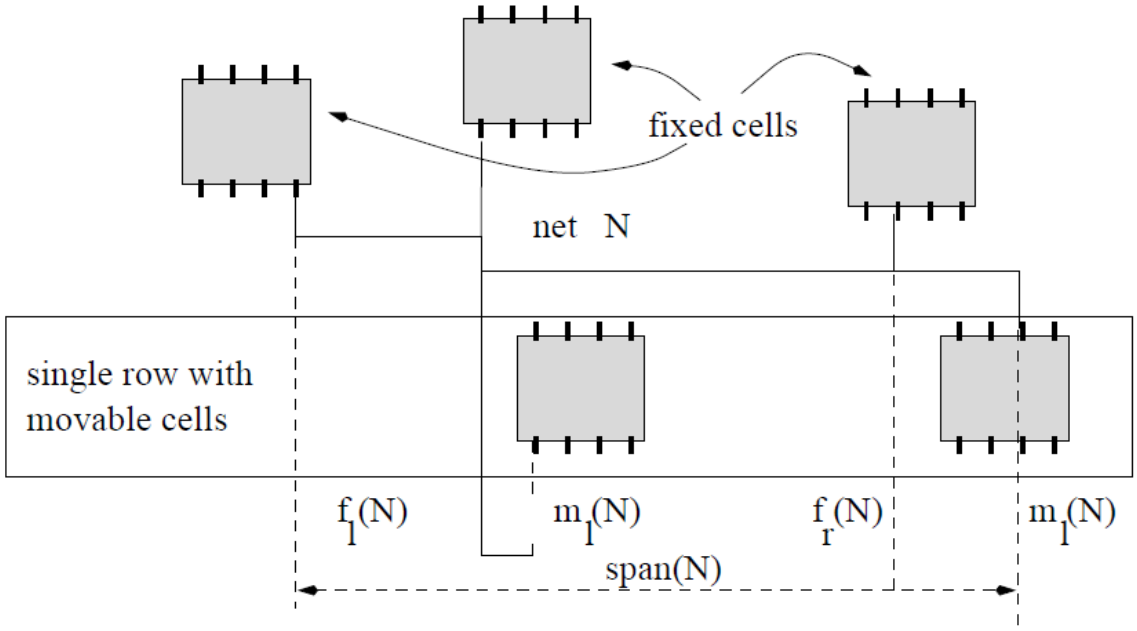
TPL Ordered Single Row (TPL-OSR) Problem

Optimize a single row of cell

Input: Ordered single row placement; pre-coloring library

Output: Legal placement and color assignment

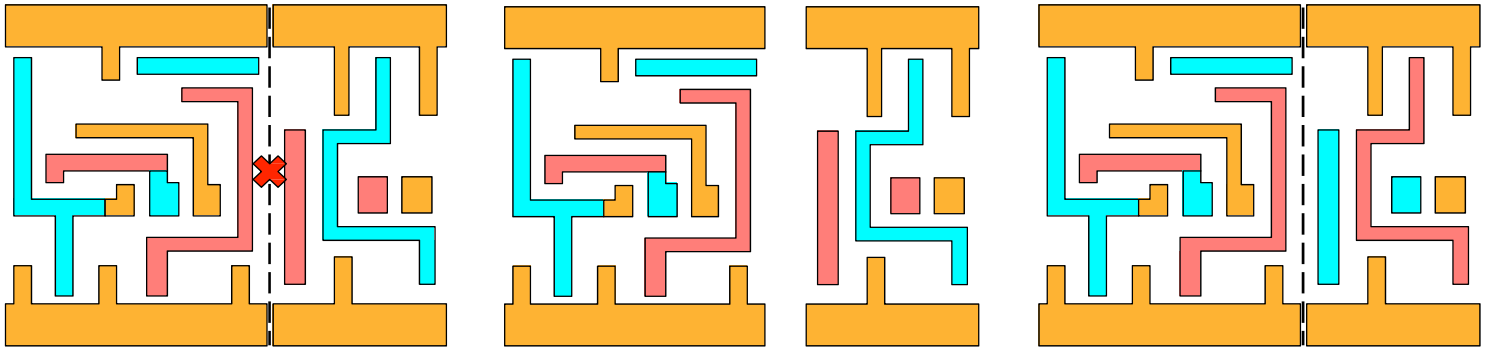
Objective: Minimize HPWL, total stitch count



When a row is optimized, we assume all other rows' cells are fixed

TPL Ordered Single Row (TPL-OSR) Problem

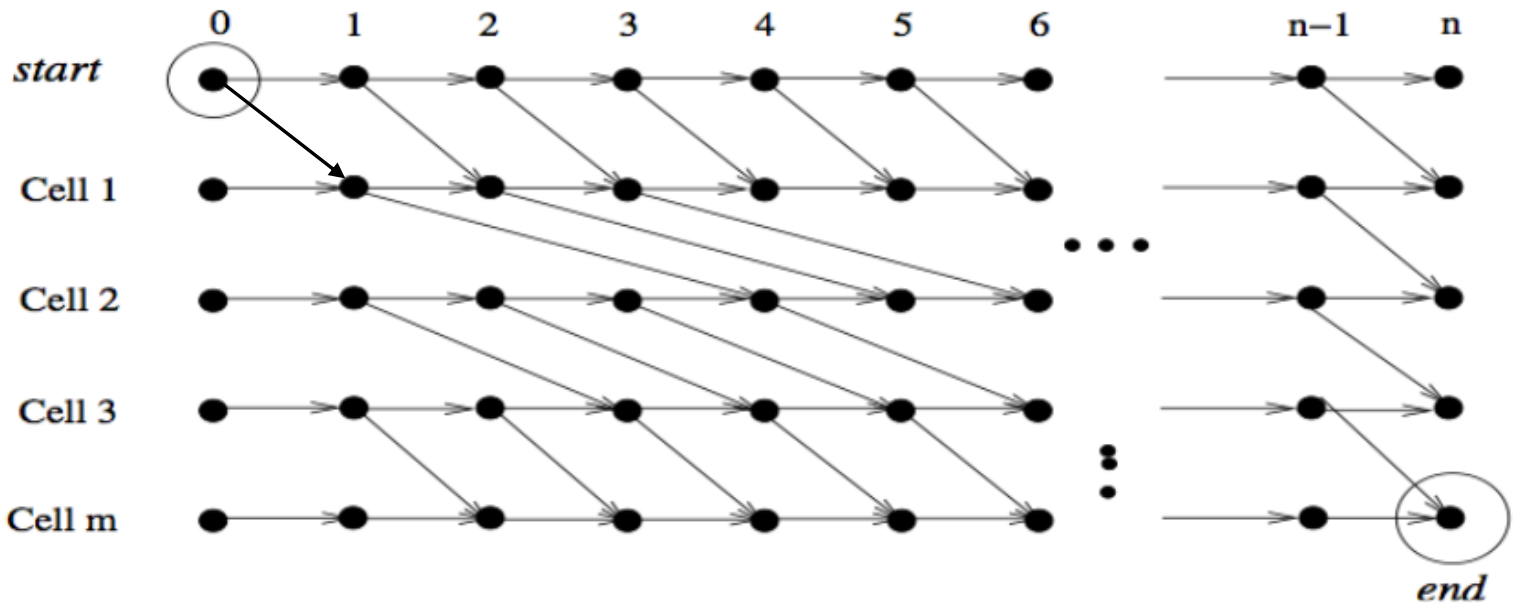
- Conflict between adjacent cells can be resolved by shifting cells or proper color assignment



Ordered Single Row Problem

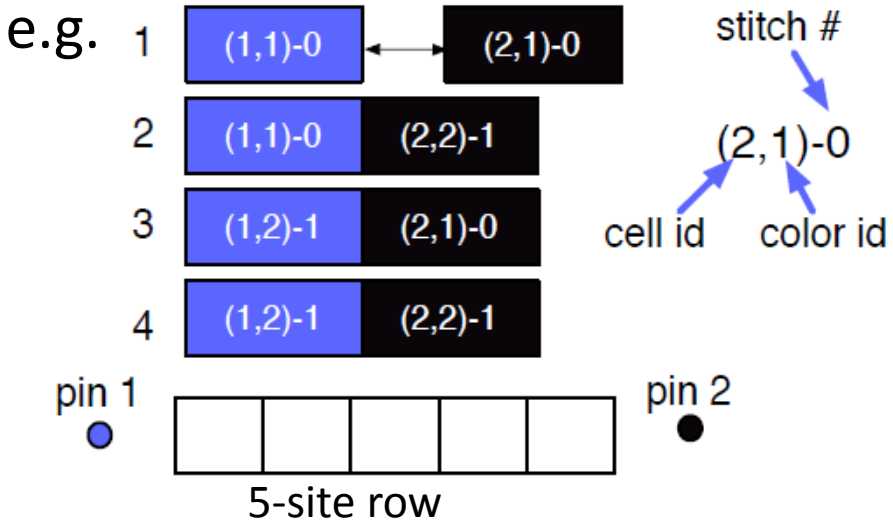
- Well studied
- Can be modelled as a shortest path problem

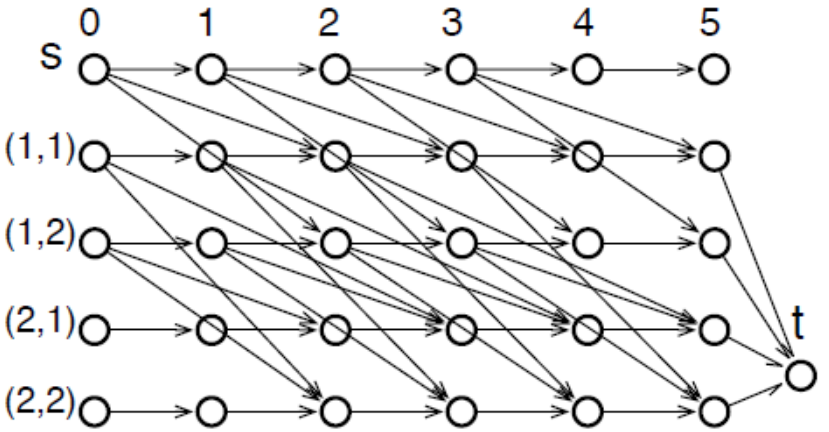
e.g. Assume there are n placement sites and widths of cells 1 to m are 1, 3, 2, …, 1



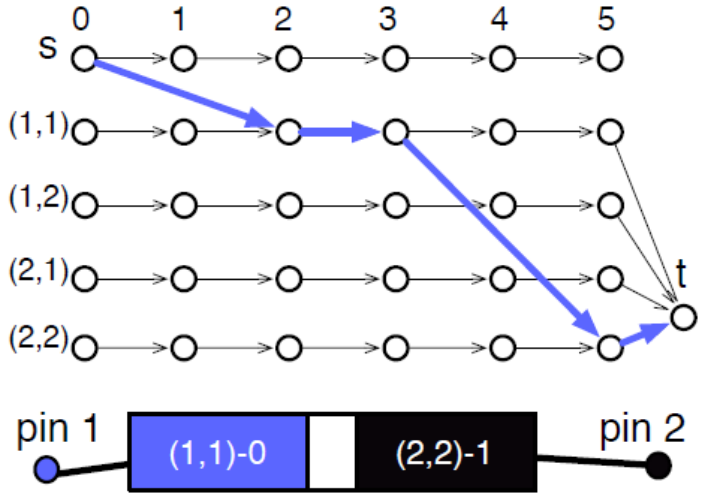
An Algorithm for TPL-OSR

Determine color assignment and cell placement simultaneously





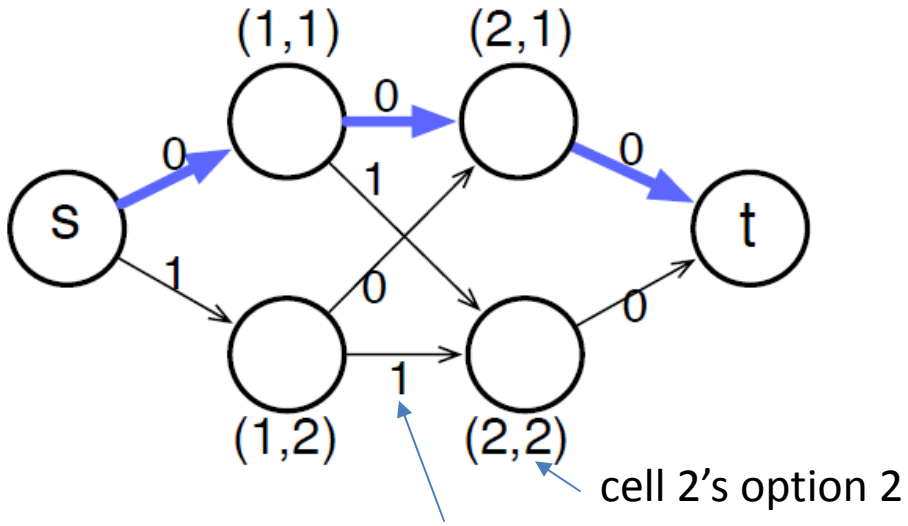
Solve as a shortest path problem



A 2-Stage Heuristic for TPL-OSR

- Faster heuristic for TPL-OSR
  - separate steps for color assignment and cell placement
- Step 1 (Color Assignment)
  - minimize stitch count and required separation
  - a shortest path problem again

e.g.

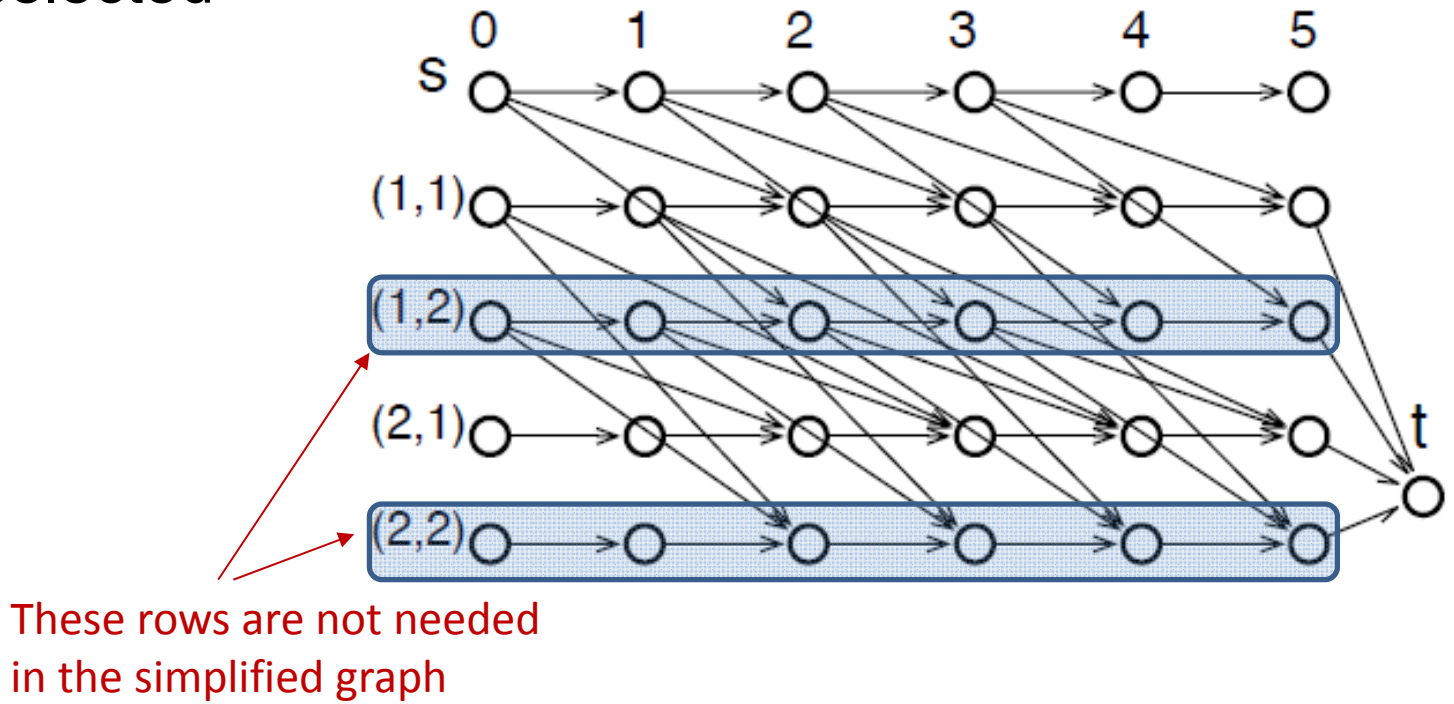


So, select cell 1's option 1 and cell 2's option 1 which incurs 0 total stitch.

A 2-Stage Heuristic for TPL-OSR

- Step 2 (Cell Placement)
  - Construct a similar graph as simultaneous approach but with just a single color option for each cell

e.g. Assume cell 1's option 1 and cell 2's option 1 have been selected



Overall Approach

TPL aware Detailed Placement

Require: cells to be placed;
  repeat
    Sort all rows;
    Label all rows as *FREE*;
    for each row row_i **do**
      Solve TPL-OSR prolbem for row_i;
      if exist unsolved cells **then**
        Global Moving; [Pan+,ICCAD'05]
        Update cell widths considering assigned colors;
        Solve traditional OSR problem for row_i;
      end if
      Label row_i as *BUSY*;
    end for
  until no significant improvement

Experiments

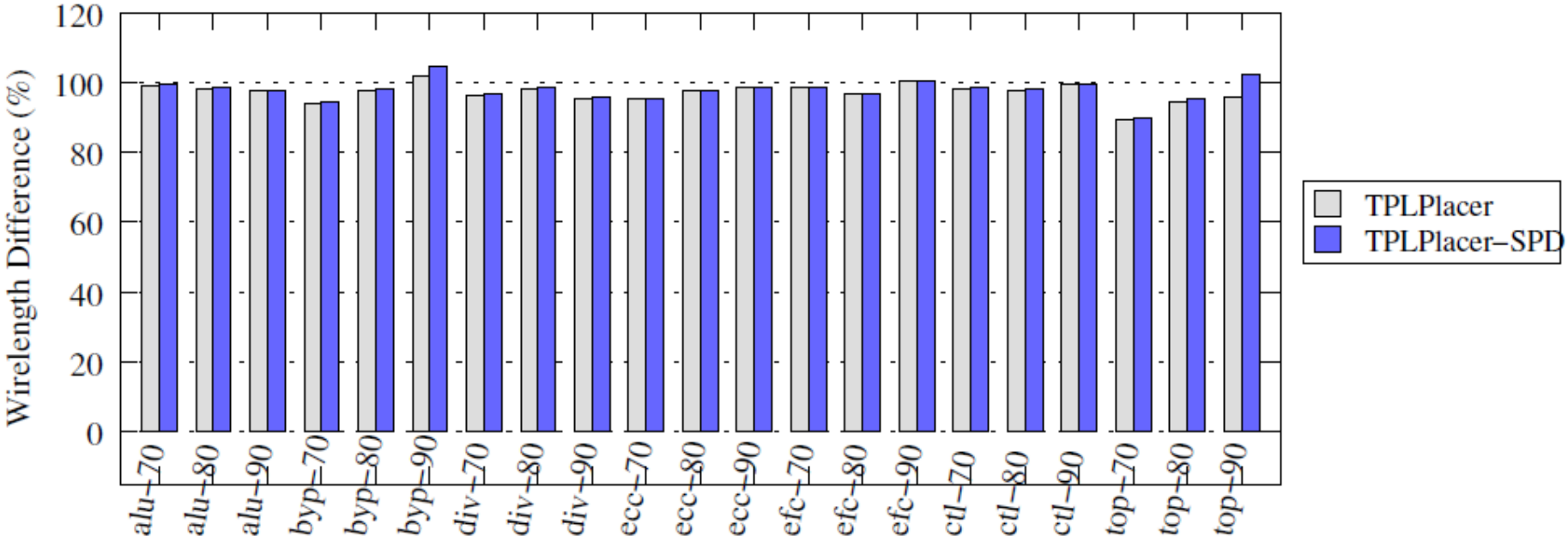
- Std-cell pre-coloring and detailed placement in C++
- Linux with 3.0GHz Intel Xeon CPU, 32GB memory
- Single thread
- Design Compiler to synthesize OpenSPARC T1 designs
- Nangate 45nm open cell library scaled to 16nm
- Encounter for initial placement results
- Three different core utilization rates: (0.7, 0.8, 0.9)

Experiments

bench	Post-Decomposition		GREEDY		TPLPlacer		TPLPlacer-SPD	
	CN#	ST#	CN#	ST#	CN#	ST#	CN#	ST#
alu-70	605	4092	0	1254	0	1013	0	994
alu-80	656	4100	N/A	N/A	0	1011	0	994
alu-90	596	3585	N/A	N/A	0	1006	0	994
byp-70	1683	9943	0	3254	0	2743	0	2545
byp-80	1918	10316	N/A	N/A	0	2889	0	2545
byp-90	2285	10790	N/A	N/A	0	3136	0	2514
div-70	1329	6017	0	2368	0	2119	0	2017
div-80	1365	5965	0	2379	0	2090	0	2017
div-90	1345	5536	0	2365	0	2080	0	2017
ecc-70	206	3852	N/A	N/A	0	247	0	228
ecc-80	265	3366	0	433	0	274	0	228
ecc-90	370	4015	N/A	N/A	0	369	0	228
efc-70	503	3333	0	1131	0	1005	0	1005
efc-80	570	4361	N/A	N/A	0	1008	0	1005
efc-90	534	4040	0	1133	0	1005	0	1005
ctl-70	425	2583	0	703	0	573	0	553
ctl-80	529	3332	0	714	0	561	0	553
ctl-90	519	3241	0	726	0	556	0	553
top-70	5893	27981	N/A	N/A	0	8069	0	8034
top-80	6775	32352	N/A	N/A	0	8120	0	8015
top-90	7313	29343	N/A	N/A	0	8710	0	7876
Average	**1700**	8664	**N/A**	N/A	**0**	2314	**0**	2186

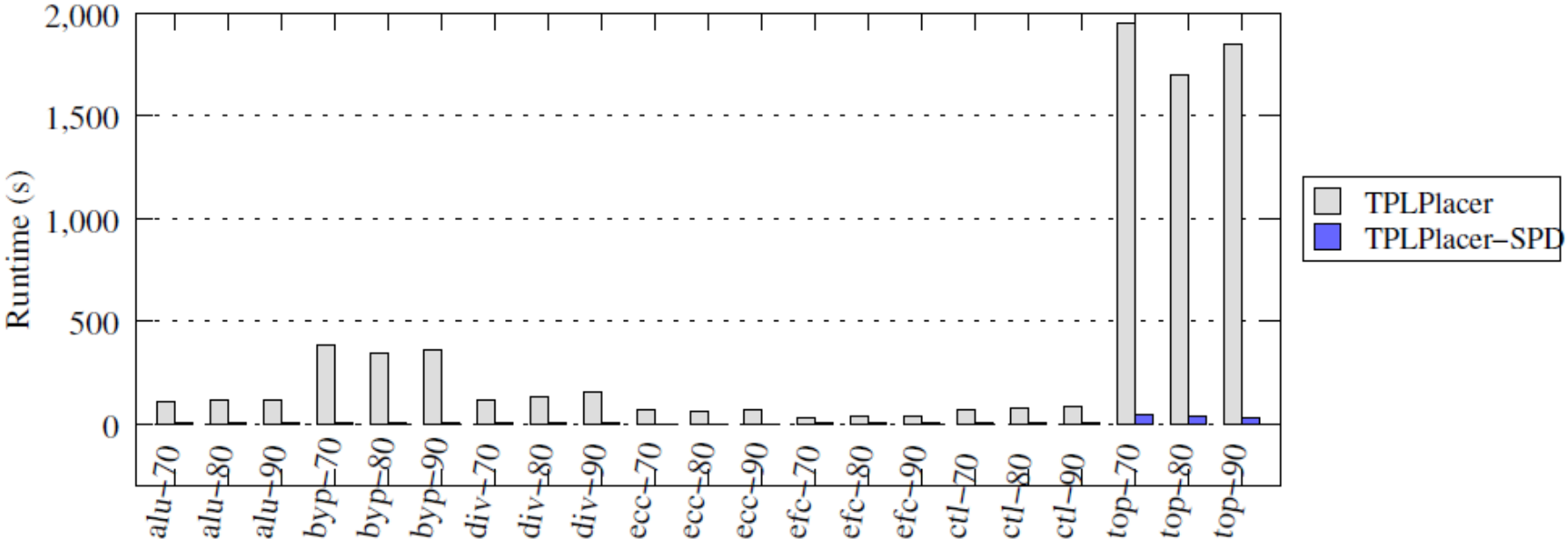
Post-Decomposition	traditional flow + layout decomposer
Greedy	greedy detailed placement algorithm [SPIE'13]
TPLPlacer	cell placement and color assignment simultaneously
TPLPlacer-SPD	fast two-stage graph models

Experiments



- 0.22% longer wirelength for TPL Placer-SPD

Experiments



- 14x speedup by TPL Placer-SPD

Reference

- [Yu+ ICCAD2013] Methodology for Standard Cell Compliance and Detailed Placement for Triple Patterning Lithography