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

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

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

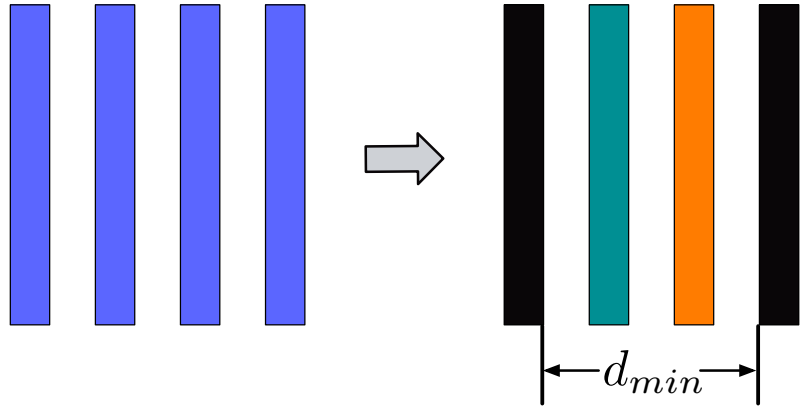
Nov. 18, 2013

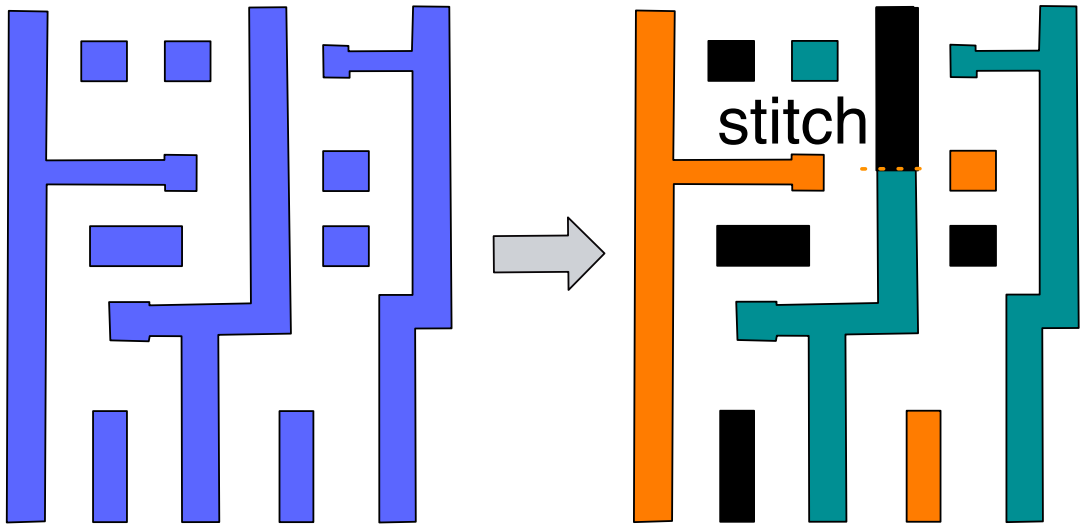
Supported by IBM scholarship, NSF, NSFC, SRC

Triple Patterning Lithography (TPL)

ITRS roadmap

- 28nm single-patterning
- 20nm double-patterning
- 14nm triple-patterning / EUV
- 10nm quadruple-patterning / EUV





TPL Layout Decomposition Works

– ILP or SAT
[Cork+,SPIE'08][Yu+,ICCAD'11][Cork+,SPIE'13]

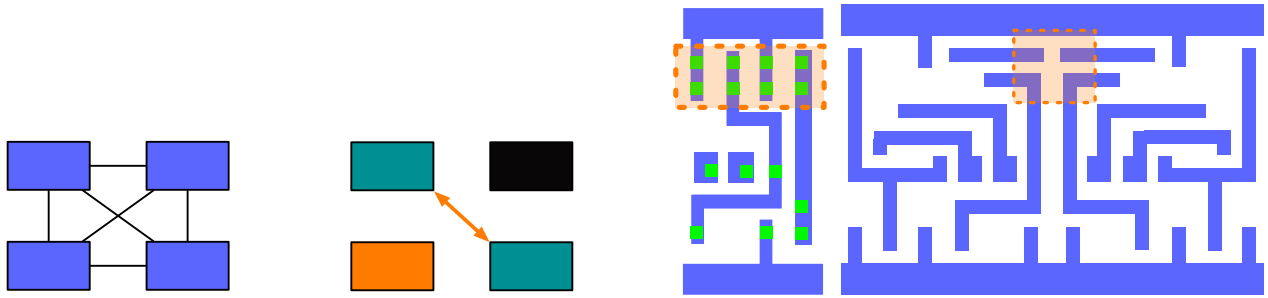
– Graph Search for Row based Layout
[Tian+, ICCAD'12][Tian+,SPIE'13][Tian+,ICCAD'13]

– Heuristic
[Ghaida+,SPIE'11][Fang+,DAC'12][Chen,ISQED'13]
[Kuang+,DAC'13][Tang+,Patent'13][Zhang+,ICCAD'13]

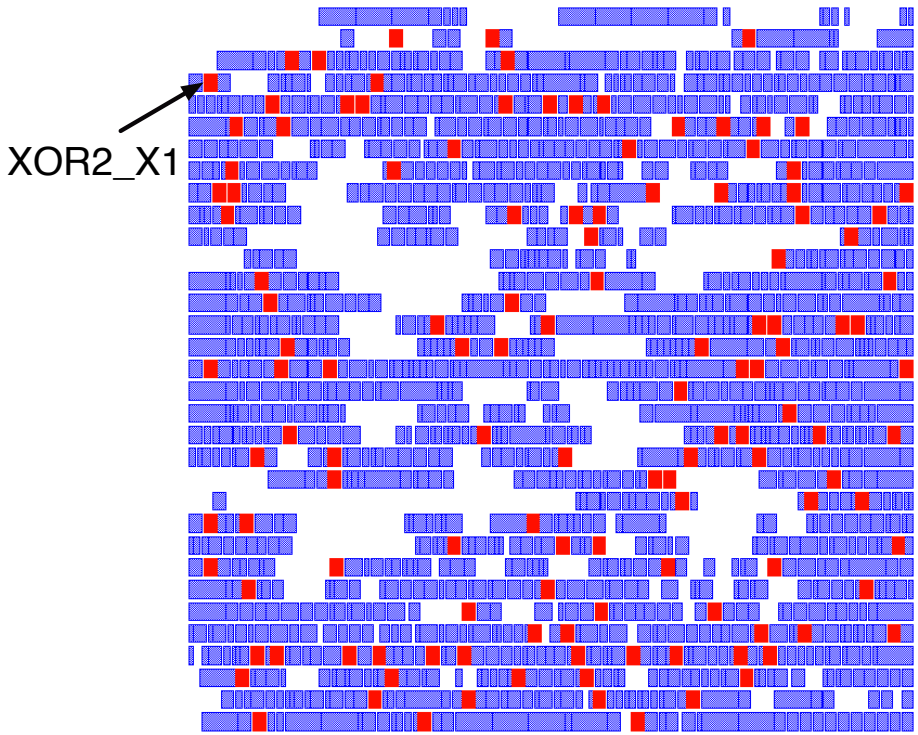
– Semidefinite Programming (SDP) (trade-off)
[Yu+, ICCAD'11][Yu+,ICCAD'13]

Post-Layout Too Late

- Native conflict from early stages



- Redundant decomposition



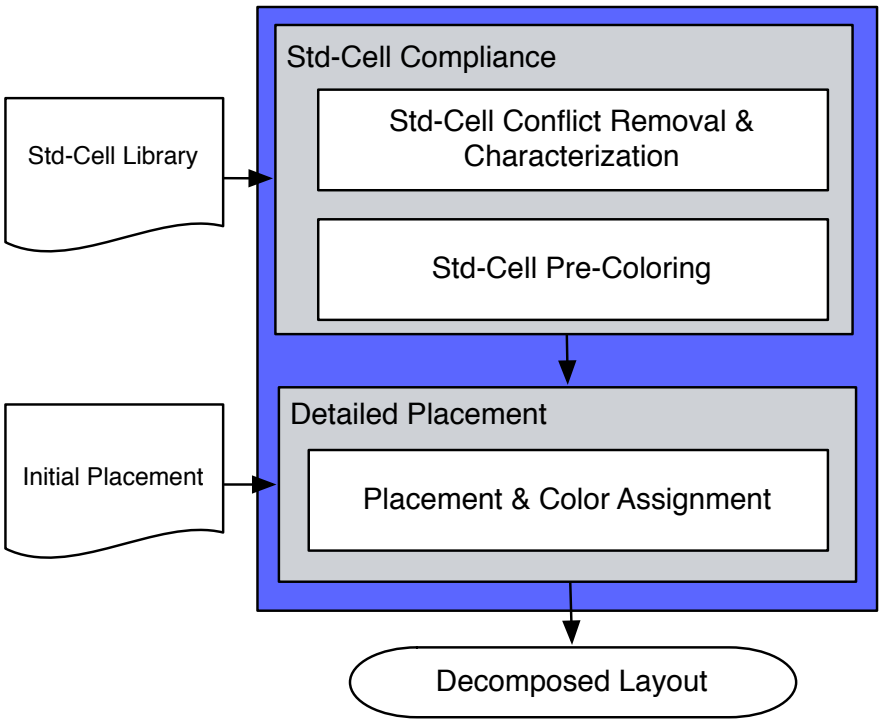
Lithography Into Early Stage

– **DFM aware Detailed Placement** [Hu+,ISPD'07] [Gupta+,ICCAD'09] [Gao+,SPIE'13] [Agarwal+,Patent'13]

– **TPL aware Routing**
[Ma+,DAC'12] [Lin+, ICCAD'12]

– **DPL aware Design Flow**
[Liebmann+,SPIE'11] [Ma+,SPIE'13]

Our TPL aware Design Flow

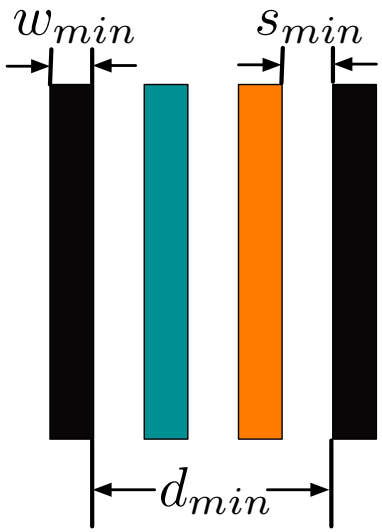


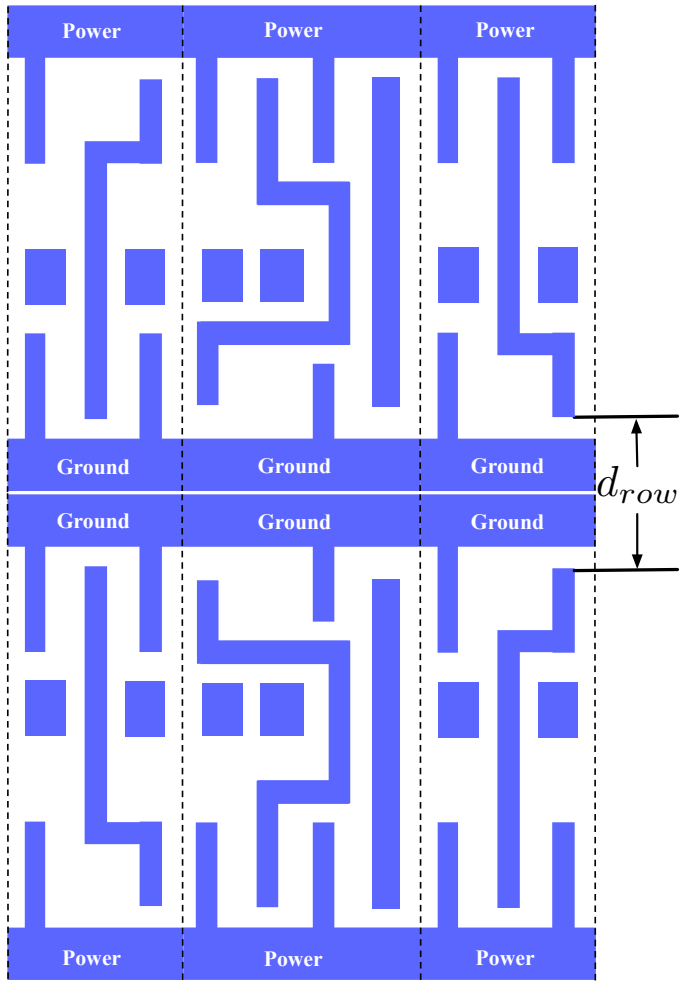
- 2 Stages
- **No** additional layout decomposition

Row Structure Layout

– d_{min}: minimum coloring distance
– d_{row}: metal spacing between rows

$$d_{min} = 2 \cdot w_{min} + 3 \cdot s_{min}$$
$$d_{row} = 4 \cdot w_{min} + 2 \cdot s_{min}$$

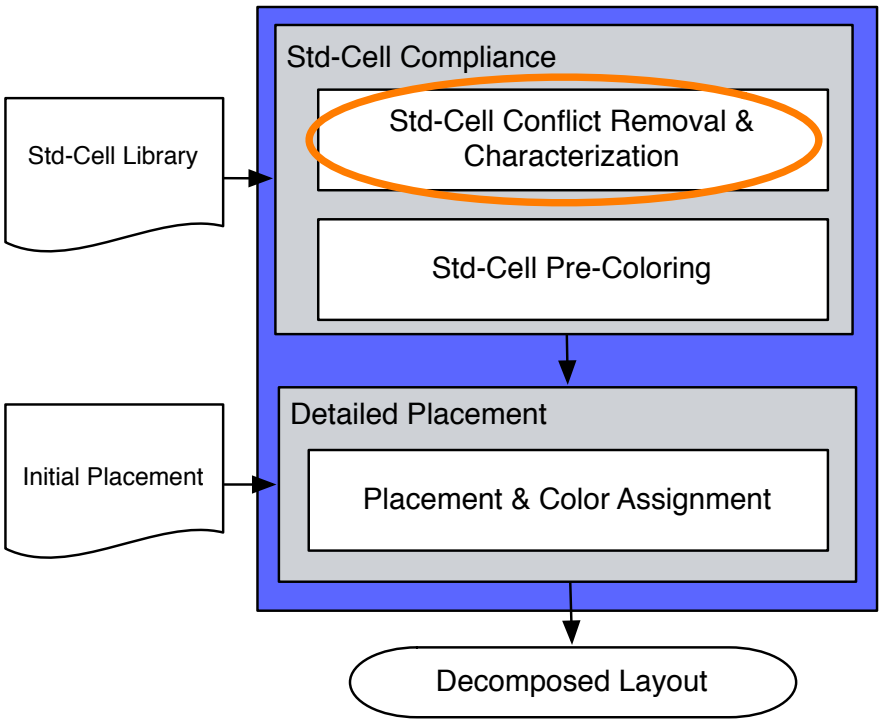


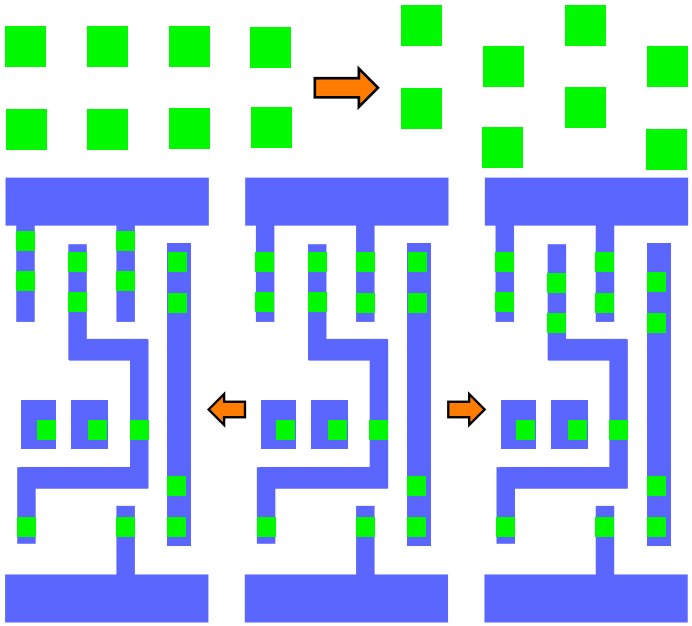


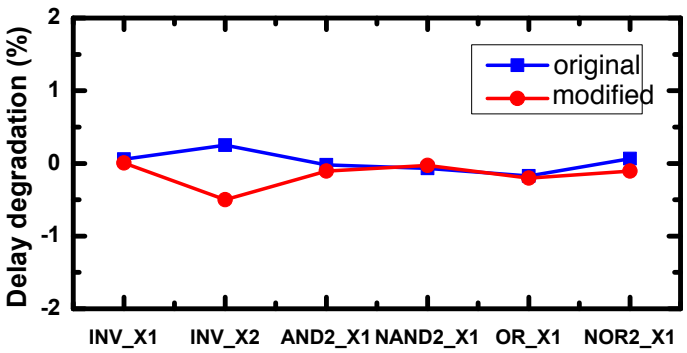
$2 \cdot w_{min} > s_{min}$, then

No interactions between rows ($d_{row} > d_{min}$).

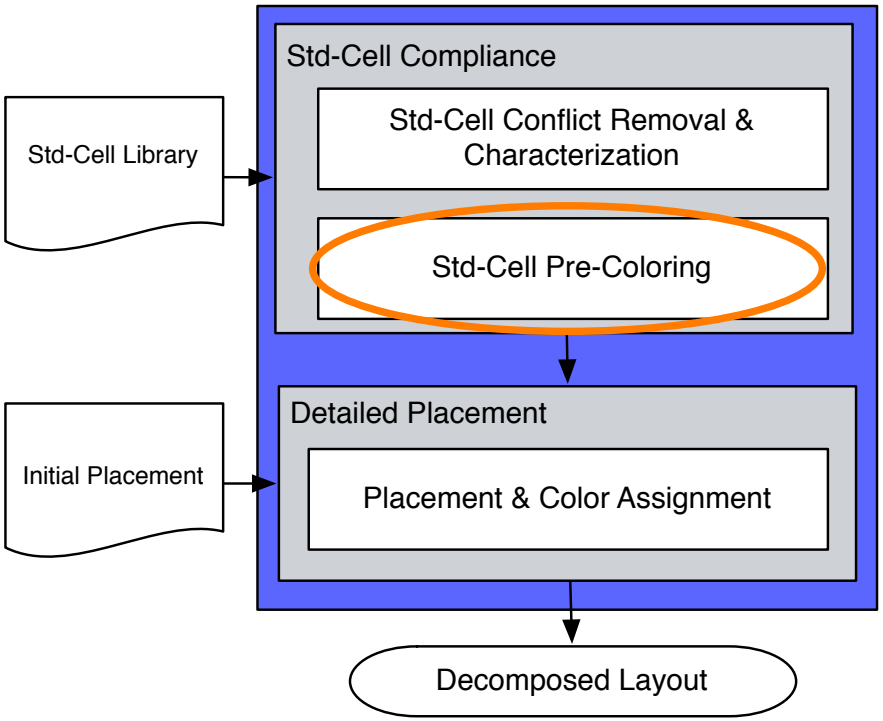
Std-Cell Conflict Removal

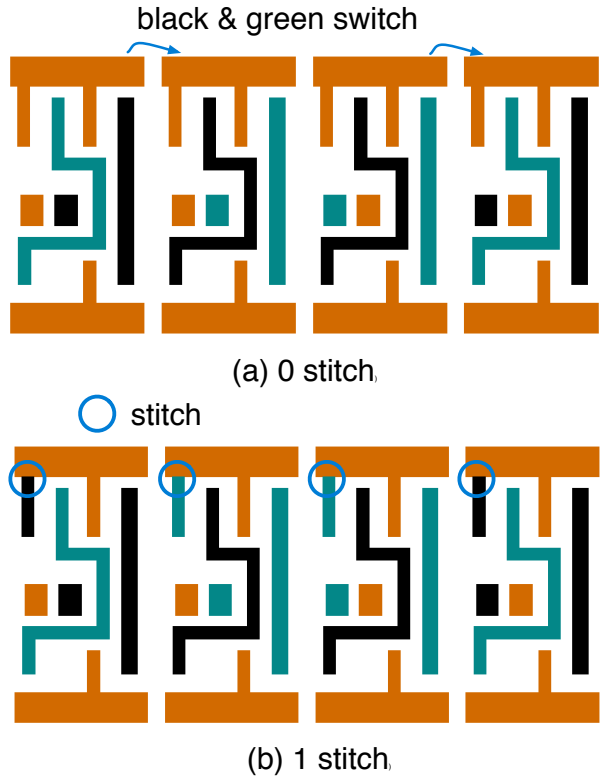


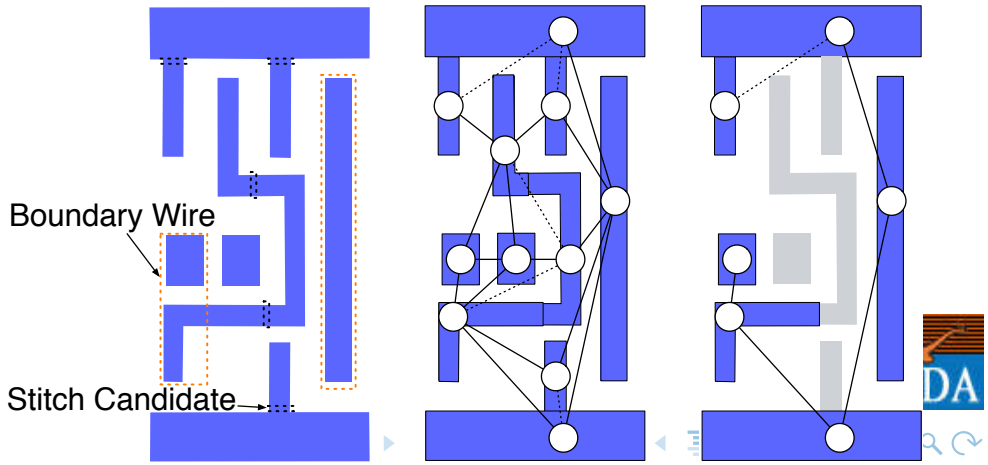




Std-Cell Pre-Coloring

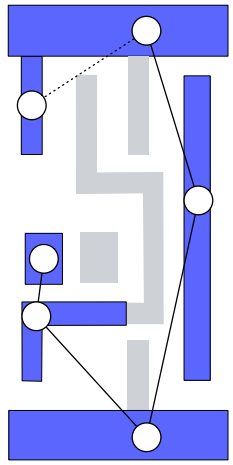


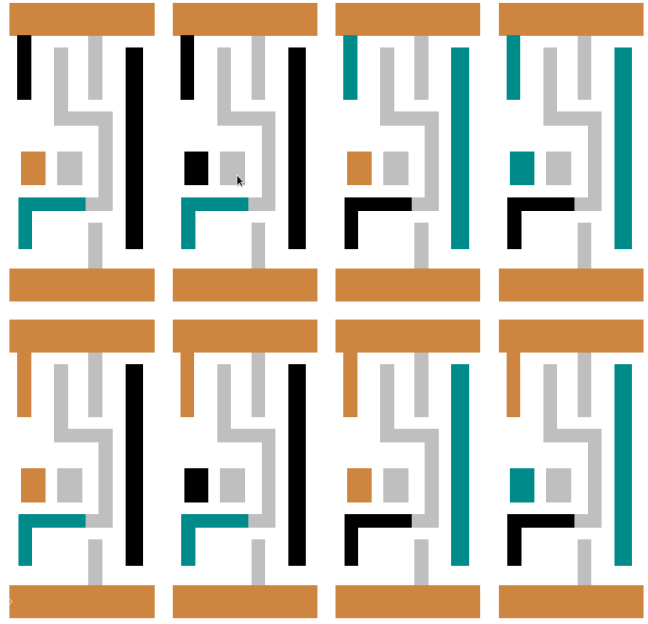




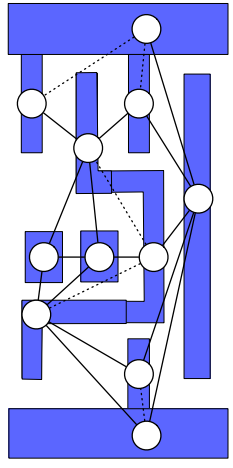
Std-Cell Pre-Coloring– Example

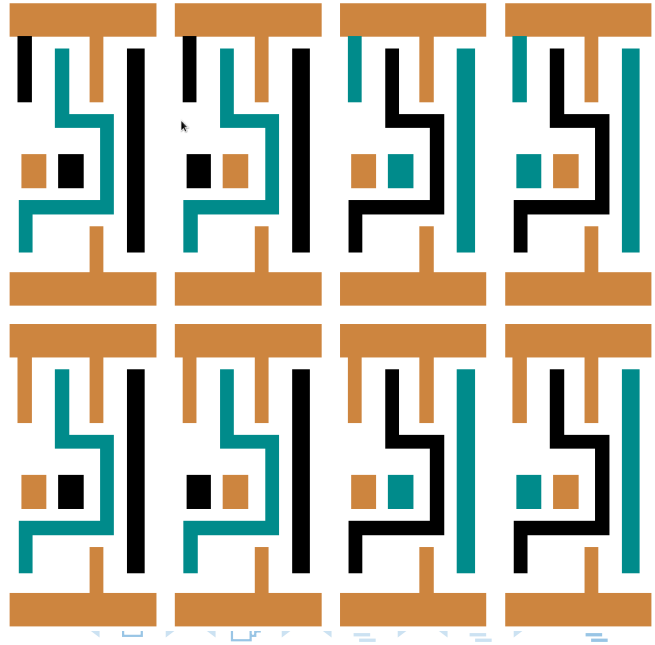
– **Stage 1:**



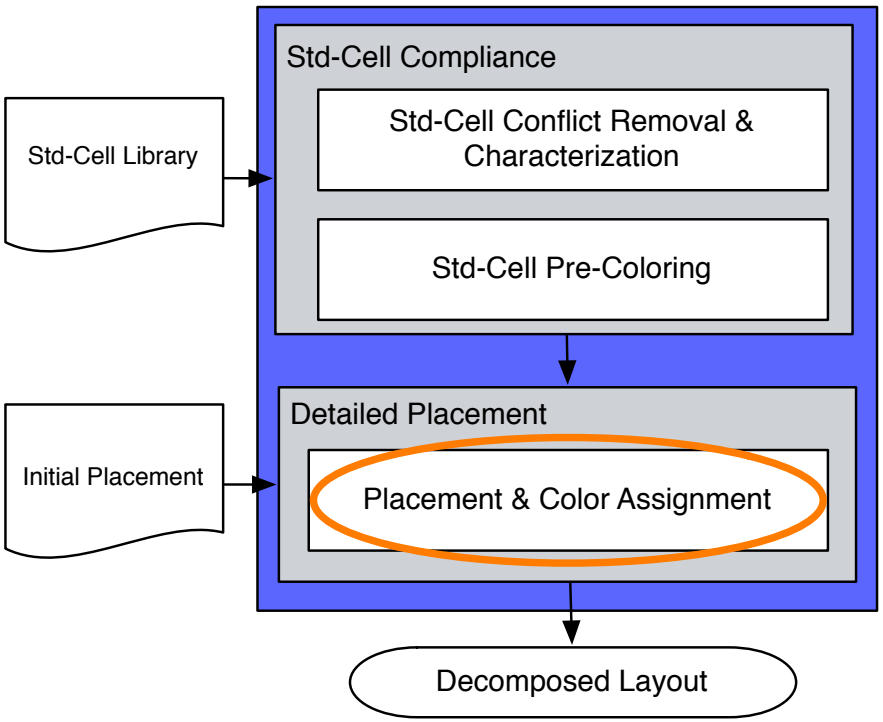


– **Stage 2:**



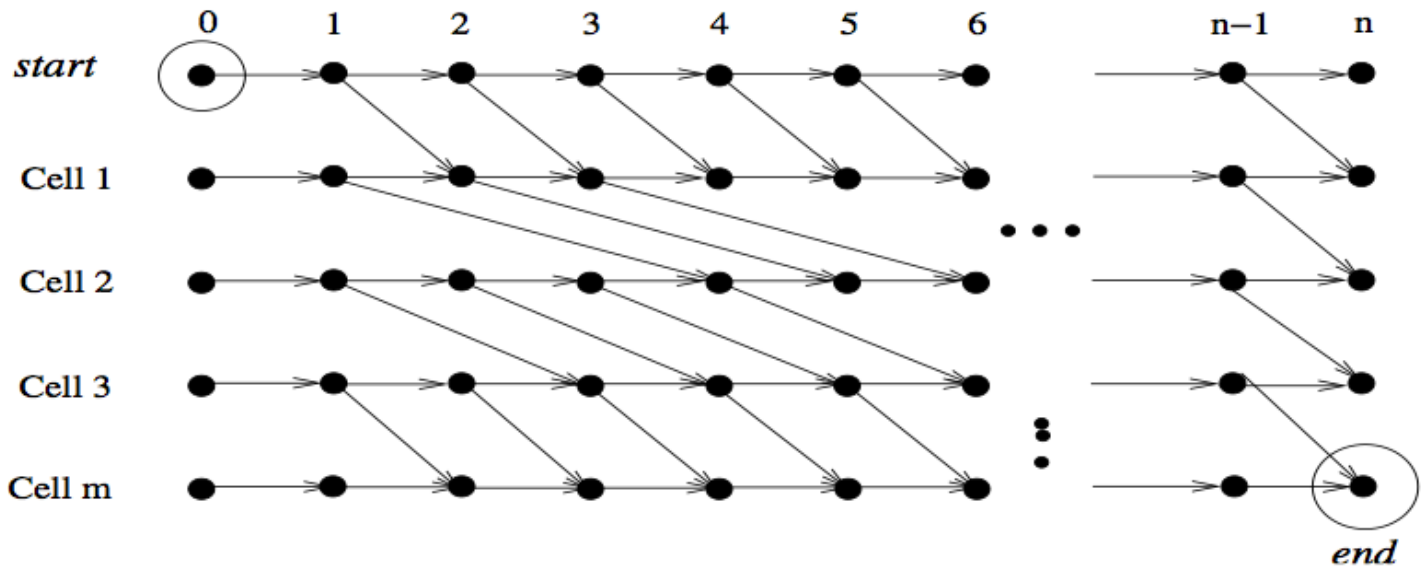


TPL aware Detailed Placement



Ordered Single Row Problem

- Well studied
- [Kahng+,ASPDAC'99] [Kahng+,ICCAD'05] [Brenner+,DATE'00]
- Shortest path based



TPL-Ordered Single Row (TPL-OSR) Problem

Problem Formulation

Input	Ordered single row placement; pre-coloring library
Output	Legal placement and color assignment
Objective	Min HPWL, total stitch number

New Challenges

- Placement + Color Assignment
- Can not estimate total row length

Graph Model for TPL-OSR

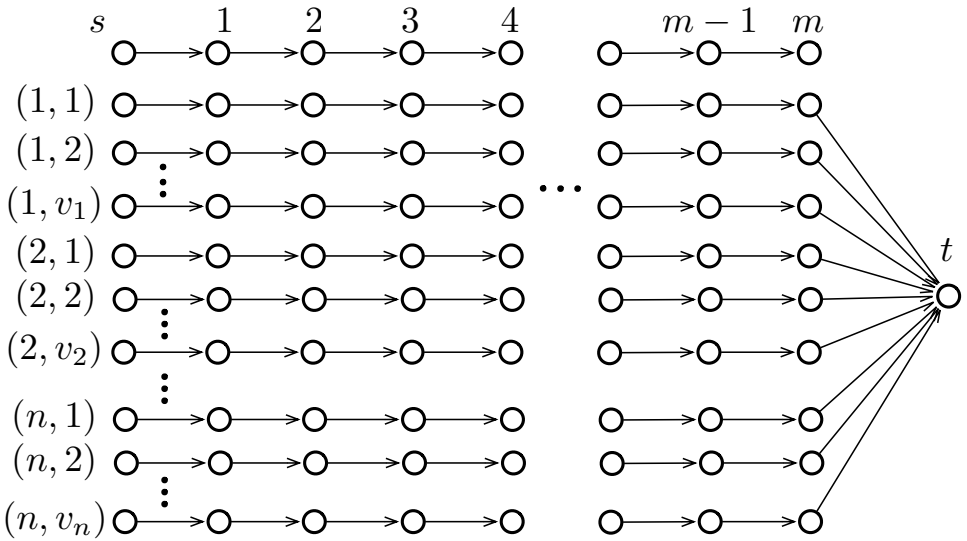


Figure : n cells to be placed in m sites (no diagonal edges shown).

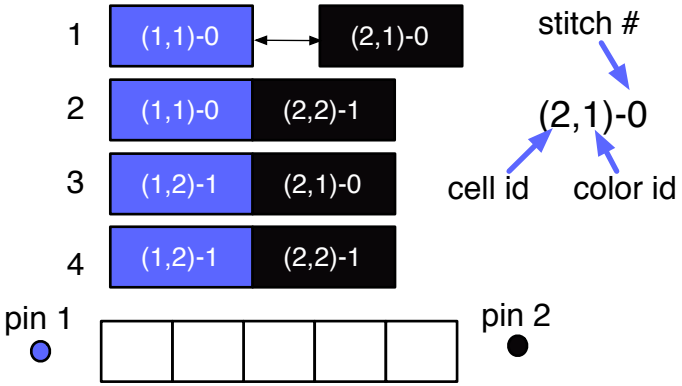
– What's New?

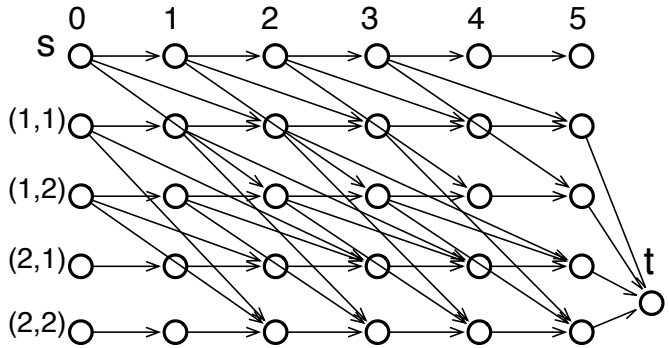
- Row $r(i,p)$: cell i is with p-th coloring solution
- Ending edges
- Cost on diagonal edges

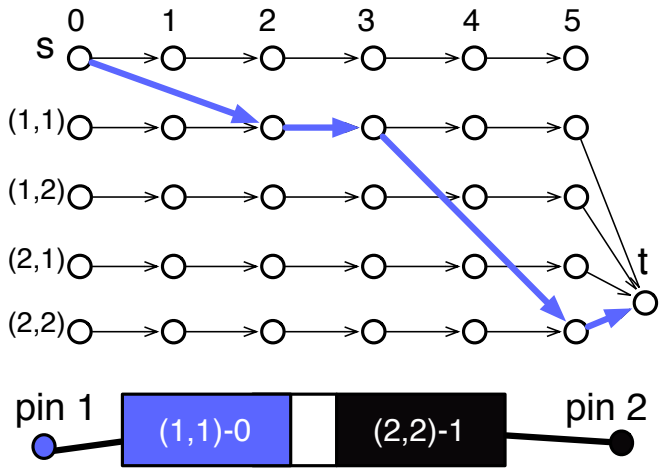
TPL-OSR solution

A shortest path from s to t, $O(nmk)$.

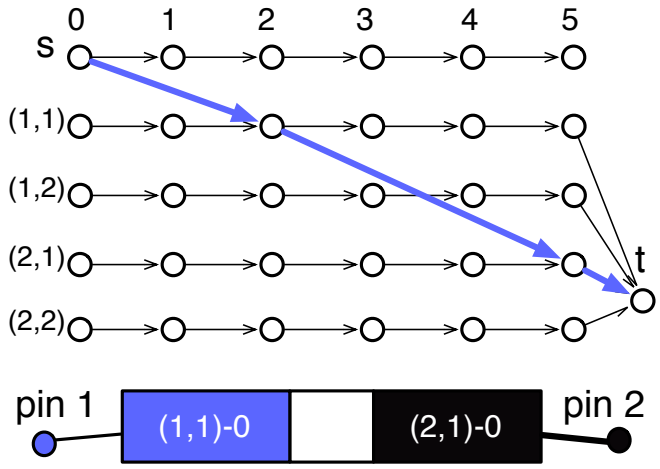
TPL-OSR Examples







(a) 1 stitch result

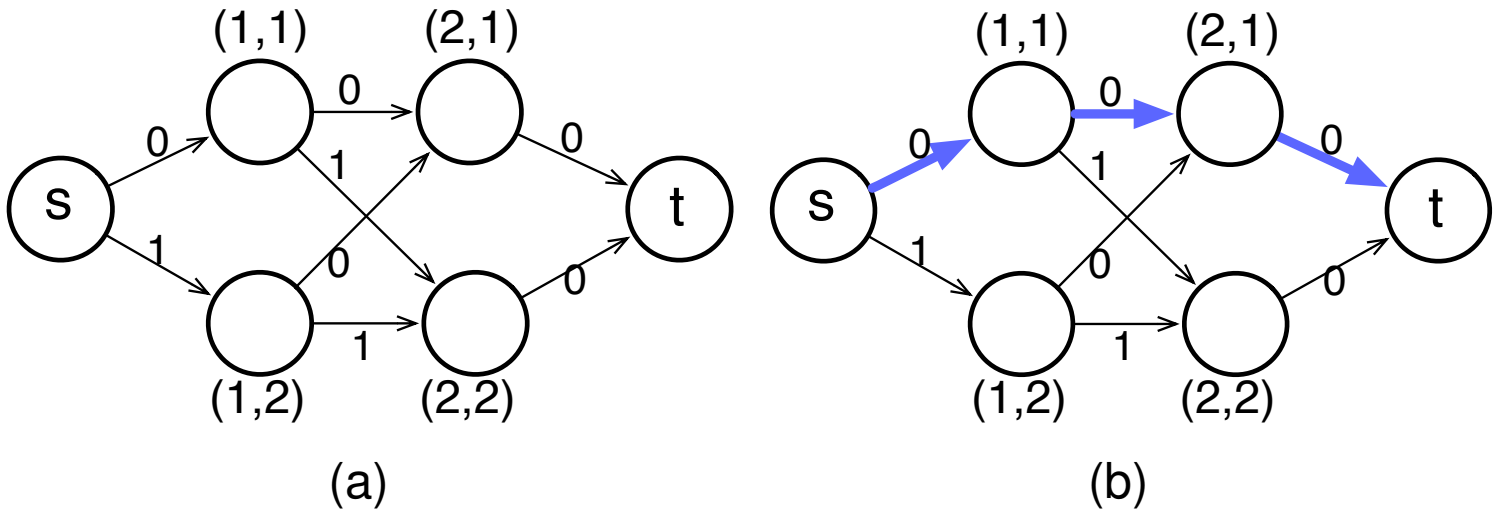


(b) 0 stitch result

Two-Stage Speedup– Stage 1

– Color assignment to minimize stich number

- $O(nk)$
- Considering current cell locations



Two-Stage Speedup– Stage 2

–Ordered single row problem to assign locations

- Coloring is fixed
- May extend cell with to resolve conflict
- traditional OSR problem
- $O(mn)$

– Speedup: $O(nmk) \rightarrow O(nk + mn)$

Overall Placement Scheme

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

Experimental Set-Up

- 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
  - alu, byp, div, ecc, efc, ctl, top
  -

	alu	byp	div	ecc	efc	ctl	top
cell#	1626	4265	2896	1303	1050	1657	12512

- Nangate 45nm open cell library scaled to 16nm
- Encounter for initial placement results
  - Three different core utilization rates: (0.7, 0.8, 0.9)

Comparison for Conflict & Stitch

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

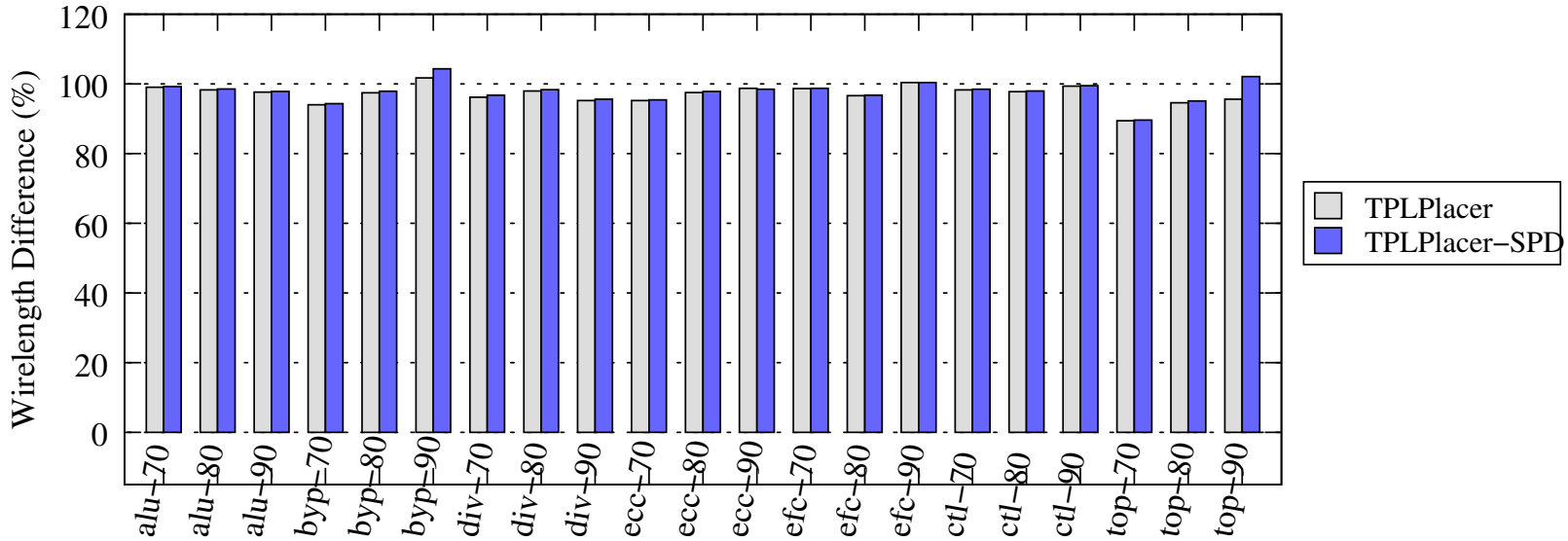
Greedy greedy detailed placement algorithm [SPIE'13]

TPLPlacer cell placement and color assignment simultaneously

TPLPlacer-SPD fast two-stage graph models

- TPLPlacer-SPD: 5% more reduction in stitches

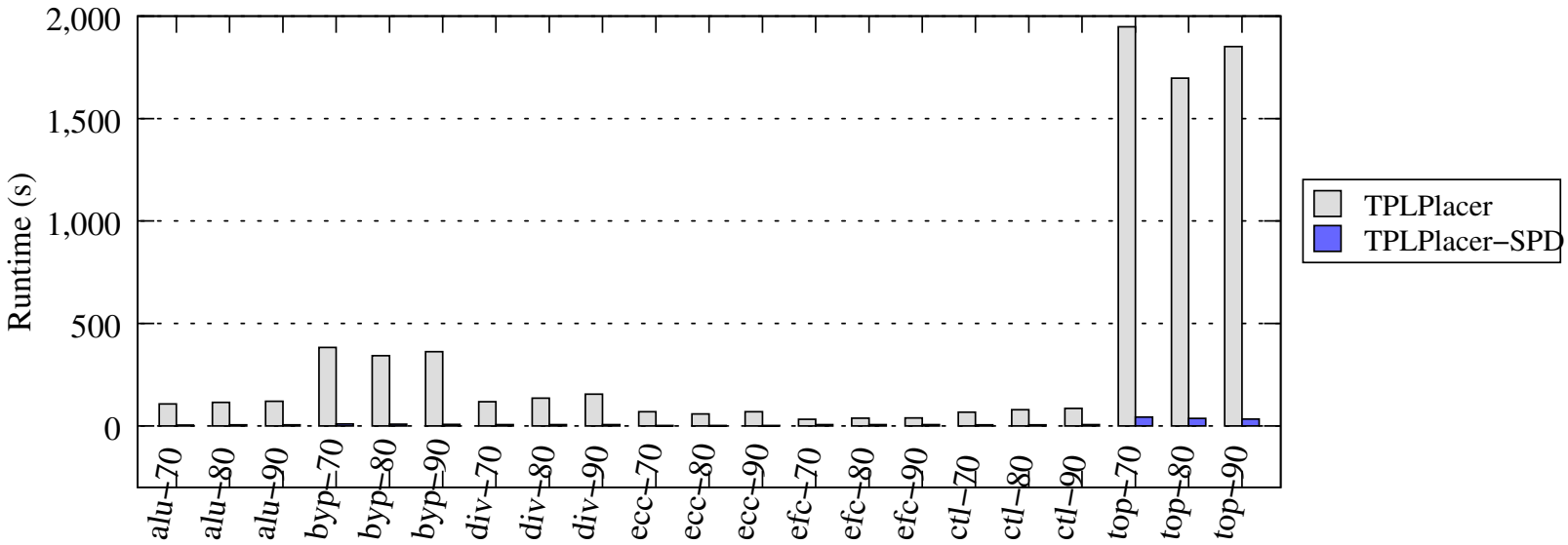
TPLPlacer-SPD v.s. TPLPlacer – Wirelength



– Wirelength

TPLPlacer-SPD : 0.22% worse

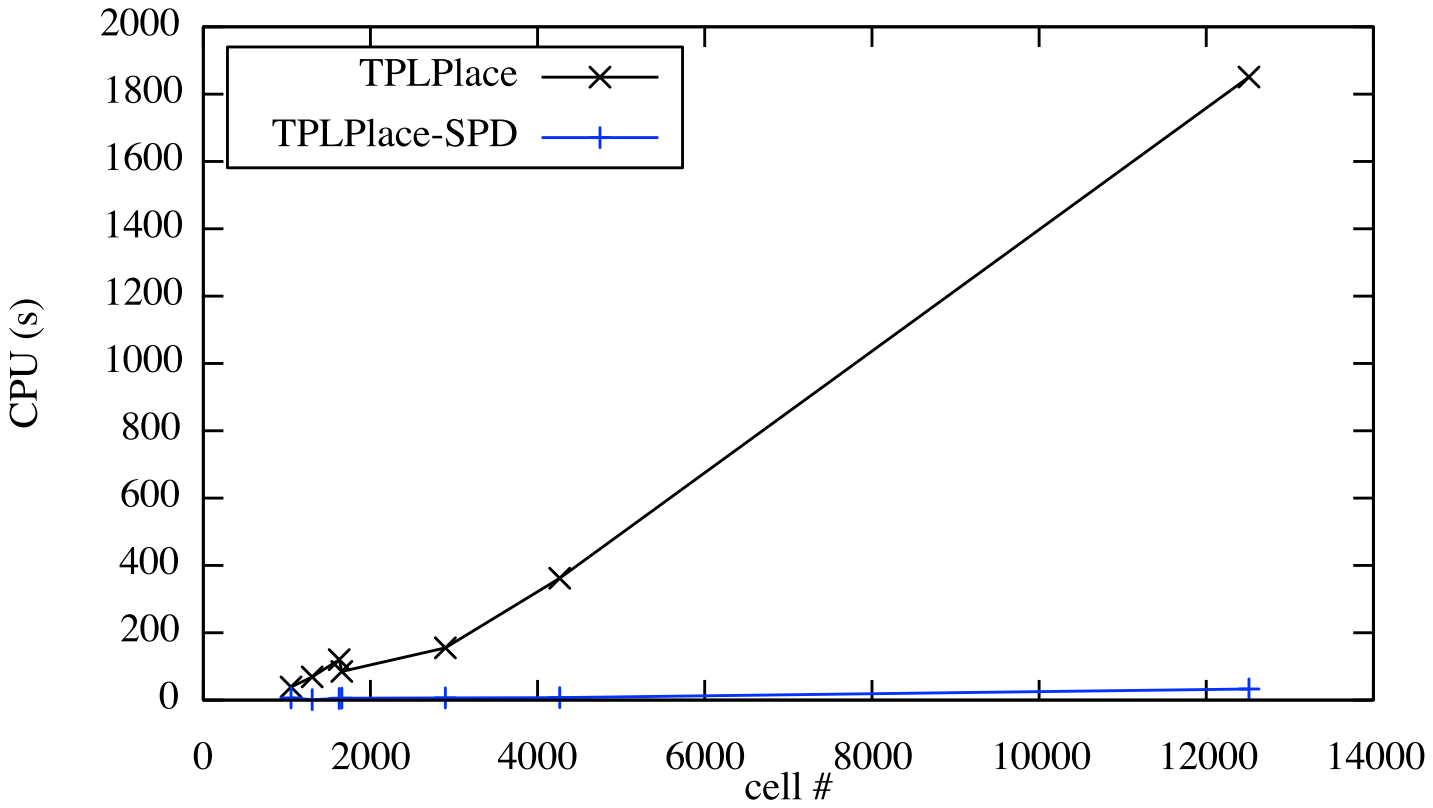
TPLPlacer-SPD v.s. TPLPlacer – Runtime



– Runtime

TPLPlacer-SPD : 14x speedup

Scalability



Conclusions and Future Work

– **Std-Cell Compliance & Detailed Placement for TPL**

– **No Just For TPL**

– **Future Work**

- Balanced density
- Congestion control in placement

Thank You !