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

<table>
<thead>
<tr>
<th>Node</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>28nm</td>
<td>single-patterning</td>
</tr>
<tr>
<td>20nm</td>
<td>double-patterning</td>
</tr>
<tr>
<td>14nm</td>
<td>triple-patterning / EUV</td>
</tr>
<tr>
<td>10nm</td>
<td>quadruple-patterning / EUV</td>
</tr>
</tbody>
</table>

\[ d_{\text{min}} \]

- \[ d_{\text{min}} \]: minimum feature size

[Diagram of lithography processes and stitch]
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

- \( d_{\text{min}} \): minimum coloring distance
- \( d_{\text{row}} \): metal spacing between rows

\[
\begin{align*}
    d_{\text{min}} &= 2 \cdot w_{\text{min}} + 3 \cdot s_{\text{min}} \\
    d_{\text{row}} &= 4 \cdot w_{\text{min}} + 2 \cdot s_{\text{min}}
\end{align*}
\]

\( 2 \cdot w_{\text{min}} > s_{\text{min}}, \) then

No interactions between rows \((d_{\text{row}} > d_{\text{min}})\).
Std-Cell Conflict Removal

- Std-Cell Library
- Initial Placement
- Std-Cell Compliance
  - Std-Cell Conflict Removal & Characterization
  - Std-Cell Pre-Coloring
- Detailed Placement
  - Placement & Color Assignment
- Decomposed Layout

Graph showing delay degradation (%) for case 1 and case 2, original vs modified.
Std-Cell Pre-Coloring

- Std-Cell Library
- Initial Placement
- Std-Cell Compliance
  - Std-Cell Conflict Removal & Characterization
  - Std-Cell Pre-Coloring
- Detailed Placement
  - Placement & Color Assignment
- Decomposed Layout

(a) 0 stitch
(b) 1 stitch

black & green switch

Stitch Candidate
Boundary Wire
Std-Cell Pre-Coloring – Example

– Stage 1:

– Stage 2:
TPL aware Detailed Placement

- Initial Placement
- Detailed Placement
  - Placement & Color Assignment
  - Decomposed Layout
  - Std-Cell Pre-Coloring
  - Std-Cell Conflict Removal & Characterization
- Std-Cell Library
Ordered Single Row Problem

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

```
start

Cell 1

Cell 2

Cell 3

Cell m

0 1 2 3 4 5 6  n-1  n

end
```
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).

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

![Graph (a)](image1)

![Graph (b)](image2)
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 \textit{FREE};
    for each row \textit{row}_i do
        Solve TPL-OSR problem for \textit{row}_i;
        if exist unsolved cells then
            Global Moving; [Pan+, ICCAD’05]
            Update cell widths considering assigned colors;
            Solve traditional OSR problem for \textit{row}_i;
        end if
        Label \textit{row}_i as \textit{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
  - 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

<table>
<thead>
<tr>
<th>bench</th>
<th>Post-Decomposition</th>
<th>GREEDY</th>
<th>TPLPlacer</th>
<th>TPLPlacer-SPD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CN#</td>
<td>ST#</td>
<td>CN#</td>
<td>ST#</td>
</tr>
<tr>
<td>alu-70</td>
<td>605</td>
<td>4092</td>
<td>0</td>
<td>1254</td>
</tr>
<tr>
<td>alu-80</td>
<td>656</td>
<td>4100</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>alu-90</td>
<td>596</td>
<td>3585</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>byp-70</td>
<td>1683</td>
<td>9943</td>
<td>0</td>
<td>3254</td>
</tr>
<tr>
<td>byp-80</td>
<td>1918</td>
<td>10316</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>byp-90</td>
<td>2285</td>
<td>10790</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>div-70</td>
<td>1329</td>
<td>6017</td>
<td>0</td>
<td>2379</td>
</tr>
<tr>
<td>div-80</td>
<td>1365</td>
<td>5965</td>
<td>0</td>
<td>2365</td>
</tr>
<tr>
<td>div-90</td>
<td>1345</td>
<td>5536</td>
<td>0</td>
<td>2365</td>
</tr>
<tr>
<td>ecc-70</td>
<td>206</td>
<td>3852</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ecc-80</td>
<td>265</td>
<td>3366</td>
<td>0</td>
<td>433</td>
</tr>
<tr>
<td>ecc-90</td>
<td>370</td>
<td>4015</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ecc-70</td>
<td>503</td>
<td>3333</td>
<td>0</td>
<td>1131</td>
</tr>
<tr>
<td>ecc-80</td>
<td>570</td>
<td>4361</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ecc-90</td>
<td>534</td>
<td>4040</td>
<td>0</td>
<td>1133</td>
</tr>
<tr>
<td>ctl-70</td>
<td>425</td>
<td>2583</td>
<td>0</td>
<td>703</td>
</tr>
<tr>
<td>ctl-80</td>
<td>529</td>
<td>3332</td>
<td>0</td>
<td>714</td>
</tr>
<tr>
<td>ctl-90</td>
<td>519</td>
<td>3241</td>
<td>0</td>
<td>726</td>
</tr>
<tr>
<td>top-70</td>
<td>5893</td>
<td>27981</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>top-80</td>
<td>6775</td>
<td>32352</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>top-90</td>
<td>7313</td>
<td>29343</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Average</td>
<td><strong>1700</strong></td>
<td><strong>8664</strong></td>
<td><strong>N/A</strong></td>
<td><strong>N/A</strong></td>
</tr>
</tbody>
</table>

**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

![Graph showing CPU (s) vs. cell # for TPLPlace and TPLPlace-SPD]
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!