Layout Pattern-driven Design Rule Evaluation

Yasmine Badr$^\alpha$, Ko-wei Ma$^\beta$, Puneet Gupta$^\alpha$

$^\alpha$EE Deptartment, UCLA

$^\beta$NVIDIA Inc.

ybadr@ucla.edu
Introduction

• **Restrictive patterning** technologies (e.g. LELE, SADP, LELELE) $\rightarrow$ non-manufacturable patterns
  – Each restrictive technology will affect routability of standard cells/design
  – Which technology to adopt?

• Sub-wavelength photolithography $\rightarrow$ **Bad Patterns**
Candidate Solutions to “Bad Patterns” Problem

- **Design Phase**: Prohibit ALL candidate bad patterns
  - *Why not?* Standard Cell Routability becomes HARDER → BIGGER area

- **Hybrid Approach**:  
  - Only prohibit selected *forbidden patterns* at Design Phase  
  - Fix the rest post-Route, in a *best effort* manner  
    - Sometimes process needs to try to allow those patterns with penalty

- **Post-Route Phase**: Allow all candidate bad patterns in design, fix them later [e.g. Legalization]
  - E.g. Flow which uses router and a pattern checker and fixer (Yang et al; SPIE 2010)
  - *Why not?* May be too late
Forbidden Patterns

• What is a good choice of patterns to forbid?
  – Highest yield-impact
    • Usually identified by lithography simulation and from failing chips data
  – Low routability-impact
    • Patterns that if forbidden:
      – don’t harshly penalize routability

→ Need an evaluation method early in the process to assess the impact of prohibiting bad patterns, as part of design rules evaluation
Forbidden Patterns

- What is a good choice of patterns to forbid?
  - Not the highest yield-impacting patterns
    - Usually identified by lithography simulation and from failing chips data
  - Low routability-impact patterns:
    - Patterns that if forbidden:
      - don’t harshly penalize routability

→ Need an evaluation method early in the process to assess the impact of prohibiting bad patterns, as part of design rules evaluation

Pattern-driven Design Rule Evaluation (Pattern-DRE)
Pattern-DRE

• Performs Pattern-aware Design Rule Evaluation
• Quick assessment of sensitivity of routability to some bad patterns ➔ select forbidden patterns
• Built on top of DRE (TCAD’12, ASPDAC’14)
Agenda

• Overview of Design Rule Evaluation Framework (DRE)
• Flow of Pattern-DRE framework
• Validation
• Sample Studies using Pattern-DRE
DRE

• A framework for early exploration of design rules, layout methodologies, and library architectures
• Standard cell-level evaluation and chip-level evaluation
• Not Pattern-aware
FLOW OF PATTERN-DRE
Flow of Pattern-DRE

1. Design Rules
2. Trans-level Netlist
3. Device-Layers Generator
4. Routing Options Generator
5. Forbidden Patterns Checker
6. Routability Metrics

- Forbidden Patterns
- Last Attempt
- Min Number of Unroutable Nets
Flow of Pattern-DRE

1. Design Rules
2. Trans-level Netlist
3. Device-Layers Generator
   - Contact Locations
4. Routing Options Generator
   - Routable?
     - Yes
       - Forbidden Patterns
     - No
       - Last Attempt
         - Yes
           - Min Number of Unroutable Nets
         - No
           - Routable?
             - Yes
               - Forbidden Patterns Checker
             - No
               - Routability Metrics

Forbidden Patterns
Routing Options Generator
Routability Metrics
Last Attempt
Min Number of Unroutable Nets
Routing Options Generator

- For each net, enumerate possible wiring solutions in the net’s bounding box
  - Use Single Trunk Steiner Tree topology

6 Wiring solutions for this net
Routing Options Generator (cont’d)

- **Enumeration of combinations** of wiring solutions of all nets → candidate routing options
  - Tree traversal
- Tree branches pruned as soon as **conflict** is found
- Conflict example:

  [Diagrams showing routing solutions]

  - Routing Solution #1: **CONFLICT** → rejected
  - Routing Solution #2: **VALID**
Tile/Pattern Representation

- Layout is represented as 2D matrix of tiles.
- Each tile/pattern is represented by
  - a segment representation [unique]
  - a node representation [necessary for conflict check]
- For a 2x2 tile:
  - Both representations are serialized as **binary strings** and saved as a number
  
  **Segment representation**
  
  ![Segment representation]
  
  => 100011010000 => 2256

  **Node representation**
  
  ![Node representation]
  
  => 1011 => 11
Conflict Detection

• A conflict occurs between wiring solutions of 2 nets if in any tile:
  – Wires overlap
    • Detected by bitwise ANDing of segments for each tile:
  – OR Wires cross
    • Detection by bitwise ANDing of nodes in the same tile
Example: AND2_X1

- 4 Nets:
  - A1 & A2:
    - 2 inputs
    - Each is a single contact net
  - ZN: output
  - Net_000
Example: AND2_X1

First wiring solution for net_000

Another wiring solution for net_000
Example: AND2_X1

First wiring solution for zn

Another wiring solution for zn
Example: AND2_X1

- Two of the several complete routing options
Flow of Pattern-DRE

Design Rules → Trans-level Netlist

Device-Layers Generator

Routing Options Generator

Forbidden Patterns Checker

Routability Metrics

Yes

Routable?

No

Contact Locations

Last Attempt

Yes

Min Number of Unroutable Nets

No

All routing options

Forbidden Patterns
2. Forbidden Patterns Checker

- **Input:**
  - list of forbidden patterns
    - Can be any size till 5 tracks x 5 tracks (currently)
  - All valid routing options
- Each *generated routing option* is checked against all *forbidden patterns*
  - Slide a window and check every formed pattern
  - If a match occurs ➔ *discard* routing option
- Very fast because of pattern representation
Flow of Pattern-DRE

1. Design Rules
2. Trans-level Netlist
3. Device-Layers Generator
4. Routing Options Generator
   - All routing options
   - Routable?
     - Yes
     - No
       - Last Attempt
         - Yes
         - No
           - Min Number of Unroutable Nets
2. Forbidden Patterns
3. Forbidden Patterns Checker
4. Routability Metrics
   - Routable?
     - Yes
     - No

Forbidden Patterns
Routing Options Generator
Device-Layers Generator
Trans-level Netlist
Design Rules
Routability Metrics

• Two Metrics reported:
  1. Number of routable cells
     • Cells which have non-zero number of routing options
  2. Total number of routing options
• Also reports number of occurrences of all patterns
How to Compare 2 Sets of Forbidden Patterns?

• Given Set A, Set B of forbidden patterns
• Run Pattern-DRE twice
  1. Set A is set of forbidden patterns
  2. Set B is set of forbidden patterns
• If **Set A has less routable cells** ➔ **Set A has higher routability impact**
• If same number of routable cells ➔ check the **total number of routing options**
  – Assume **Set A has smaller** number
    ➔ **harder** to route the cells without patterns of Set A
    ➔ Less chance of successful **post-route fix** for rest of patterns
    ➔ **Set A has higher routability impact**
Flow of Pattern-DRE

Design Rules → Trans-level Netlist → Device-Layers Generator → Routing Options Generator

Forbidden Patterns Checker → Routability Metrics

Routable? Yes/No

Last Attempt Yes/No

Min Number of Unroutable Nets

DRE → Flow of Pattern-DRE

Forbidden Patterns → Routeable?

All routing options

Contact Locations

Yes/No

No
Minimum Number of Unroutable Nets

• The routing options generator may fail to find a conflict-free routing option for the cell.
• Objective: find the routing solution with minimum number of unrouted nets
• Formulated and solved as ILP.
VALIDATION, EXPERIMENTS & RESULTS
Validation

• **Device-layer generation:**
  – Less than 2% average error in area in comparison to **Nangate Open Cell Library**
  – 38 minutes for entire library on single CPU

• **Routing estimation**
  – 12% higher wire-length on average and 44x faster in comparison to FLUTE Steiner-tree router (C. Chu et al; TCAD 2008)

• **Pattern Counting**
  – Patterns that contribute to ~82.4% in Nangate layouts, take up ~81.5% of counts in our approach
  – Cosine Similarity = 0.86
    • Measured for 2 vectors of pattern counts Nangate vs. PatternDRE
Metrics Index

- **Routing Options**: Total number of valid non-forbidden routing options of all cells
- **Routable cells**: Number of cells that have non-zero number of routing options
Experiment #1: SADP vs. LELE

- **Objective**: how much routability do we sacrifice for better overlay control?
- For **SADP**: assume trim not allowed to create any edges (no overlay sensitive edges)
  - Most of the patterns that are SADP-compliant are LELE-compliant
  - Some patterns are considered LELE-compliant but not SADP-compliant

![Diagram with LELE and SADP symbols]
Experiment #1: SADP vs. LELE (cont’d)

• Samples of forbidden patterns (258 patterns)

• Disclaimer: for proper conclusions, enumerate all SADP – incompatible patterns that are allowed by LELE
Experiment #1: SADP vs. LELE (cont’d)

- **SADP**: with forbidden patterns
- **LELE**: without any forbidden patterns

<table>
<thead>
<tr>
<th></th>
<th>Routable Cells</th>
<th>Routing Options</th>
<th>Change in Routing Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>SADP</td>
<td>77</td>
<td>2766</td>
<td>-17%</td>
</tr>
<tr>
<td>LELE</td>
<td>78</td>
<td>3338</td>
<td></td>
</tr>
</tbody>
</table>

- Sacrifice 1 routable cell and 17% of routing options for **better overlay control**
Experiment #2: LELE vs. EUVL

• Forbidden patterns:
  – LELE:
    • Patterns of size 4x4
    • Enumerated then found LELE-incompliant using commercial DP decomposer
  – EUVL: none

<table>
<thead>
<tr>
<th></th>
<th>Routable Cells</th>
<th>Routing Options</th>
<th>Decrease in Routing Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>LELE</td>
<td>72</td>
<td>1440</td>
<td>56.9%</td>
</tr>
<tr>
<td>EUVL</td>
<td>78</td>
<td>3338</td>
<td></td>
</tr>
</tbody>
</table>

• By using LELE instead of the unconstrained EUVL, we sacrifice routability of 7.8% of the cells, and 56.9% of the routing options.
Experiment #3: Diffusion Location

- **Objective**: compare two front-end choices for location of diffusion area:
  - Close to power rails
  - Close to P/N interface

<table>
<thead>
<tr>
<th>Diffusion Location</th>
<th>Routable Cells</th>
<th>Routing Options</th>
<th>Decrease in Routing Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close to Power rail</td>
<td>78</td>
<td>2772</td>
<td></td>
</tr>
<tr>
<td>Close to P/N interface</td>
<td>74</td>
<td>861</td>
<td>6.9%</td>
</tr>
</tbody>
</table>
Conclusion

- Proposed Pattern-aware Design Rule Evaluation framework
- Can be used to assess the implications of certain restrictive technologies, or blocking bad patterns

Future Work

- Integrate with a lithography simulator to consider yield-severity of patterns
QUESTIONS?
Backup
Device-layers Generator

Pairing → Folding → Chaining/Stacking → Ordering

Transistor pair → Large transistor → Folded transistor → Transistor stack

ICCAD’11, TCAD’12
1. Routing Options Generator (cont’d)

• If bounding box of the net has skewed aspect ratio ➔ long wiring in one direction
  – Ignore routing solutions with trunk in that direction

• On-track routing