Layout Pattern-driven Design Rule Evaluation

Yasmine Badr[α], Ko-wei Ma[β], Puneet Gupta[α]

[α]*EE Deptartment, UCLA*

[β]*NVIDIA Inc.*

ybadr@ucla.edu

Introduction

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

- Sub-wavelength photolithography→ **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 c of bid?
  - No

Pattern-driven Design Rule Evaluation (Pattern-DRE)

➔ process to as tterns, as part of de

Pattern-DRE

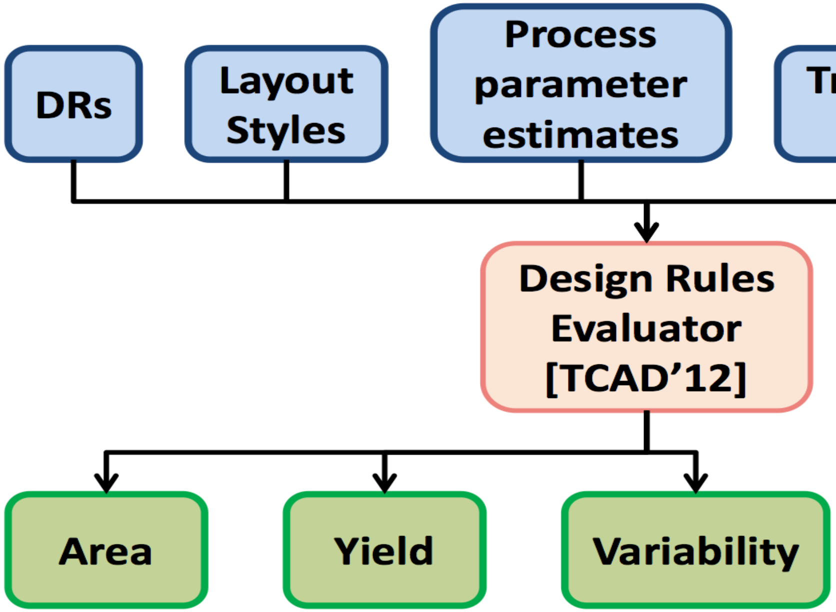
- Performs Pattern-aware Design Rule Evaluation
- Quick assessment of **sensitivity** of routability to some **bad patterns**→ select forbidden patterns
- Built on top of **DRE** *(TCAD'12, ASPDAC'14)*

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



DRs

Layout Styles

Process parameter estimates

Trans-level Netlist

Benchmark Designs

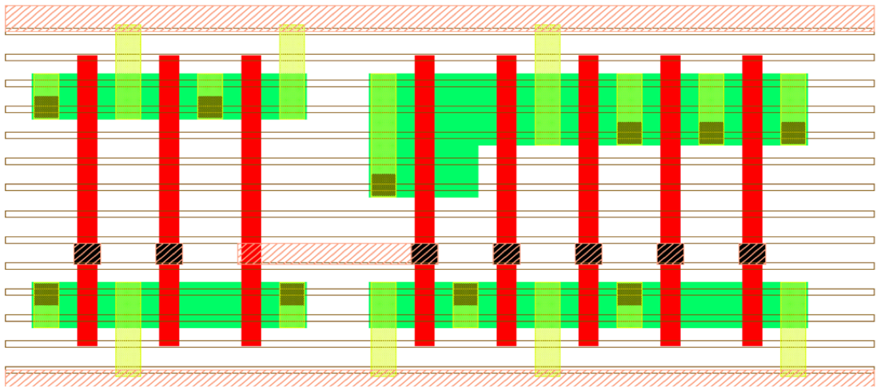
Design Rules Evaluator [TCAD'12]

Area

Yield

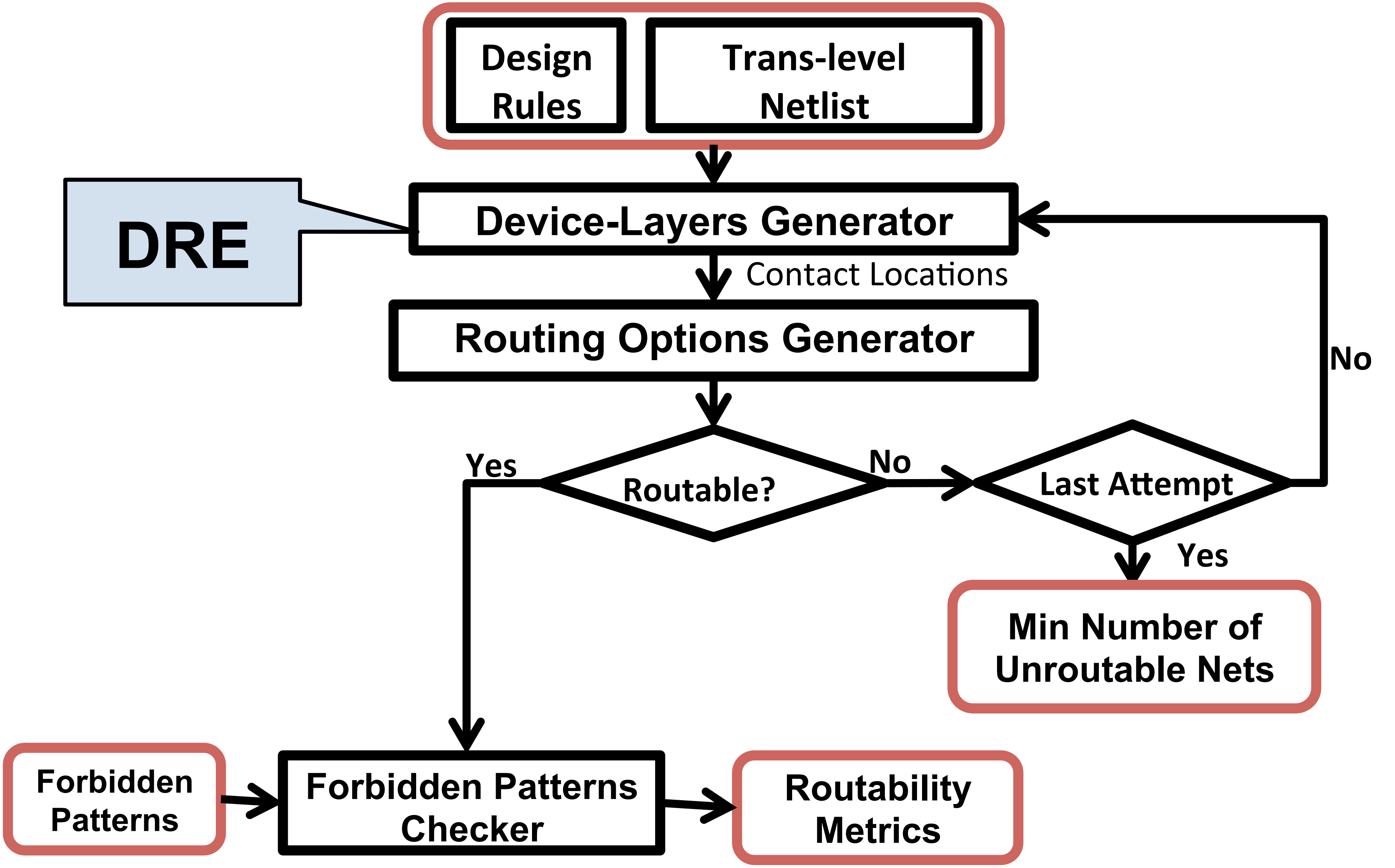
Variability

Half-adder layout

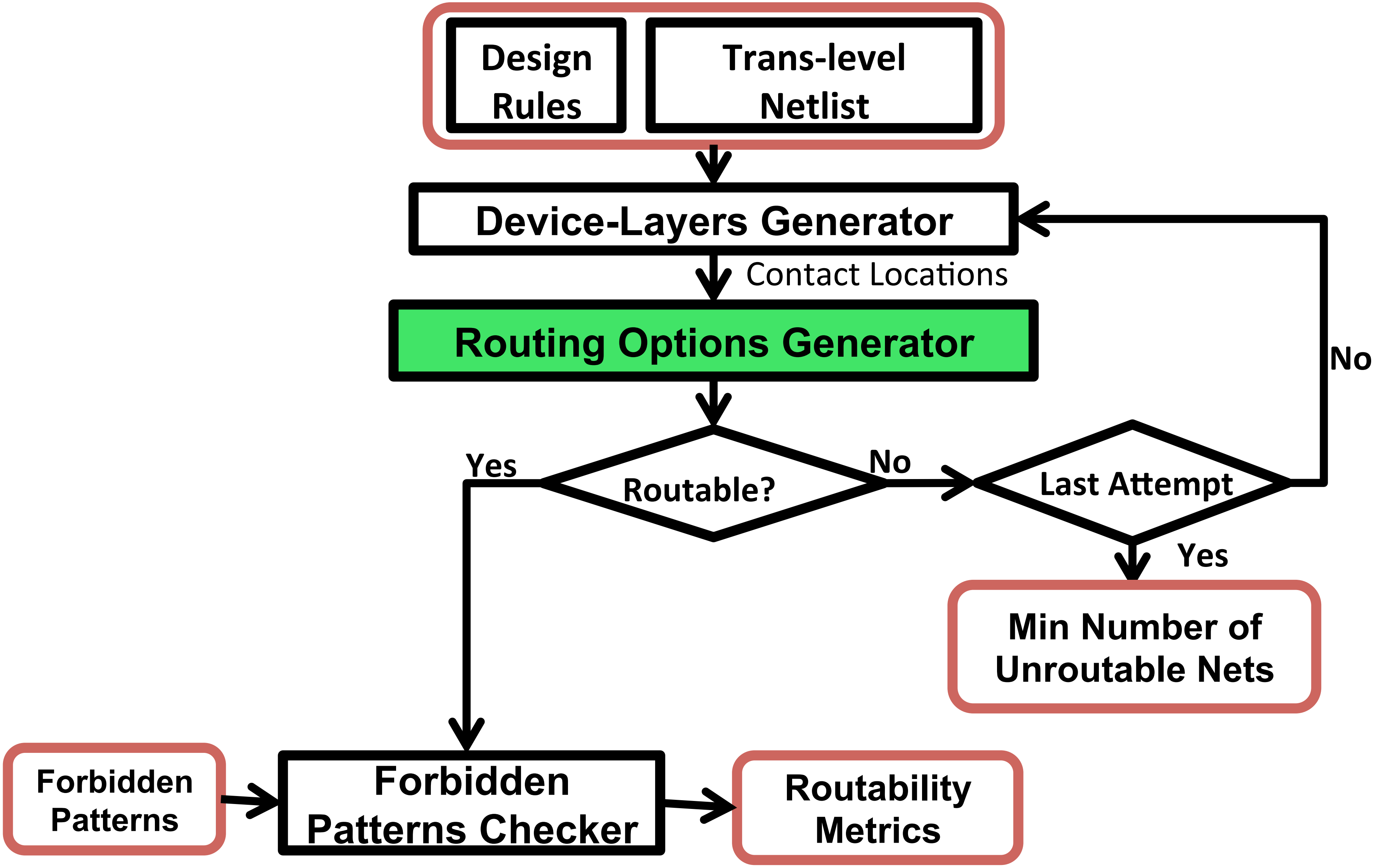


FLOW OF PATTERN-DRE

Flow of Pattern-DRE



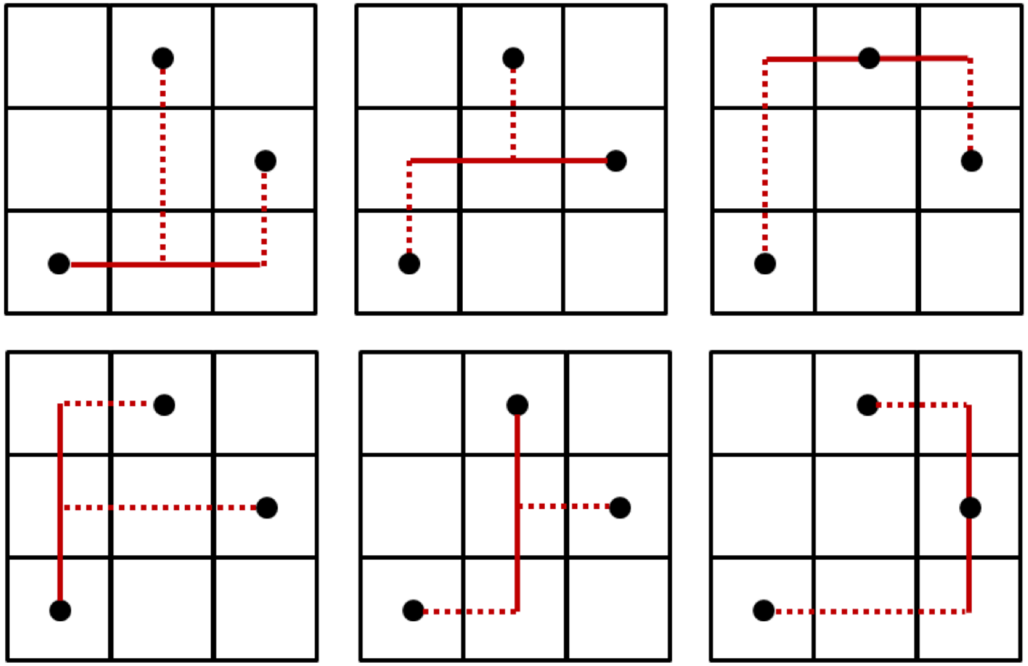
Flow of Pattern-DRE



Routing Options Generator

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

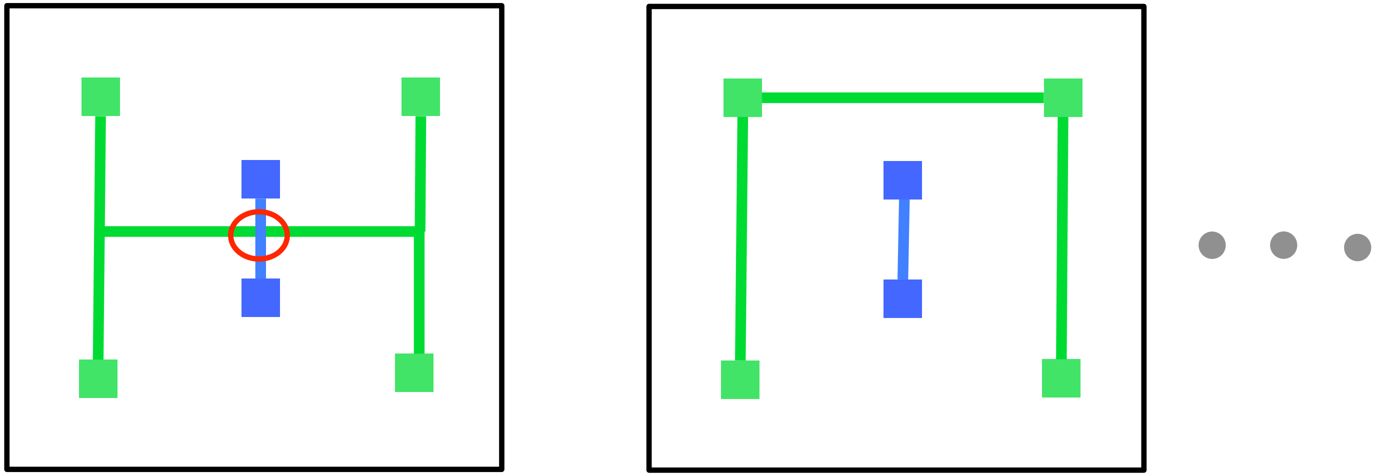
6 Wiring solutions for this net



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:

🟦 Contact for Net#1 🟩 Contact for Net#2



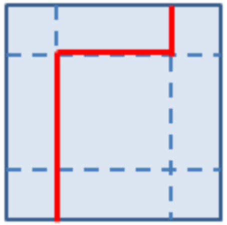
Routing Solution #1:
CONFLICT→ rejected

Routing Solution #2:
VALID

Tile/Pattern Representation

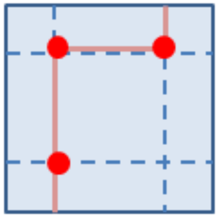
- Layout is represented as 2D matrix of tiles.
- Each tile/pattern is represented by
  - a segment representation [unique]
  - a node representation [necessary for conflict check]
- For a 2x2 tile:

Segment representation



=> 100011010000 => 2256

Node representation

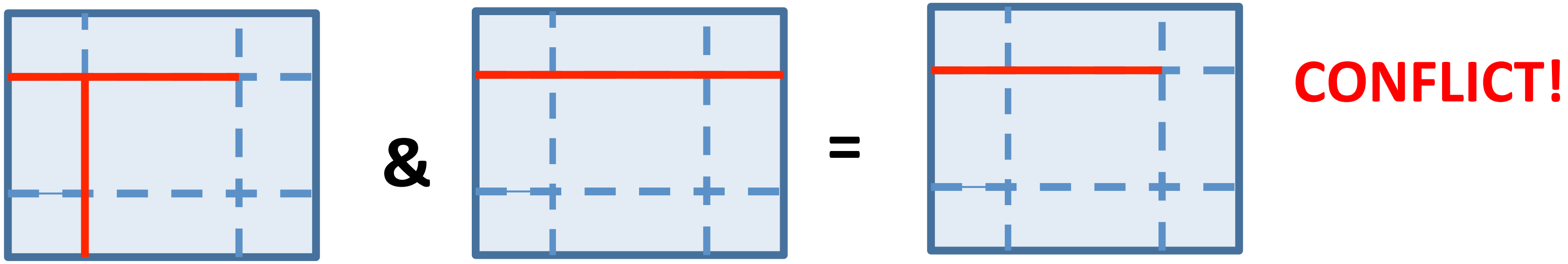


=> 1011 => 11

- Both representations are serialized as **binary strings** and saved as a number

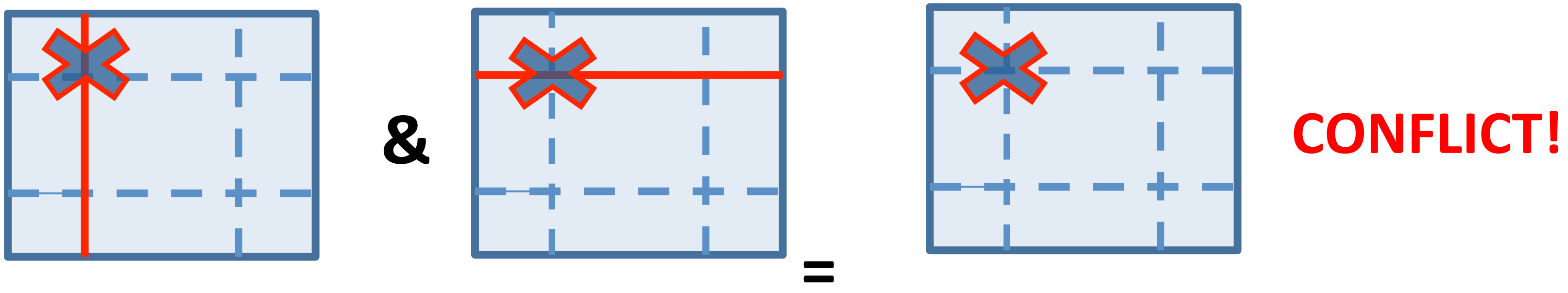
Conflict Detection

- A conflict occurs between wiring solutions of 2 nets if in any tile :
  - Wires overlap
    - Detected by bitwise ANDing of segments for each tile:



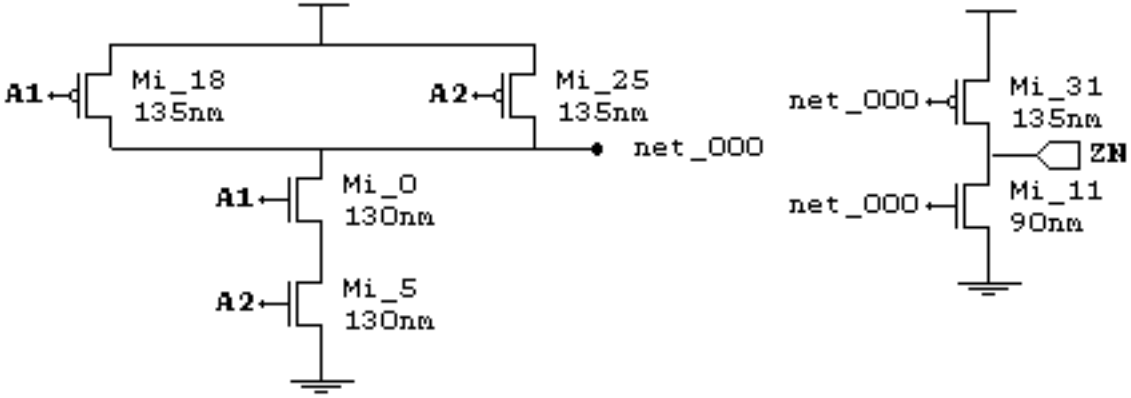
& = CONFLICT!

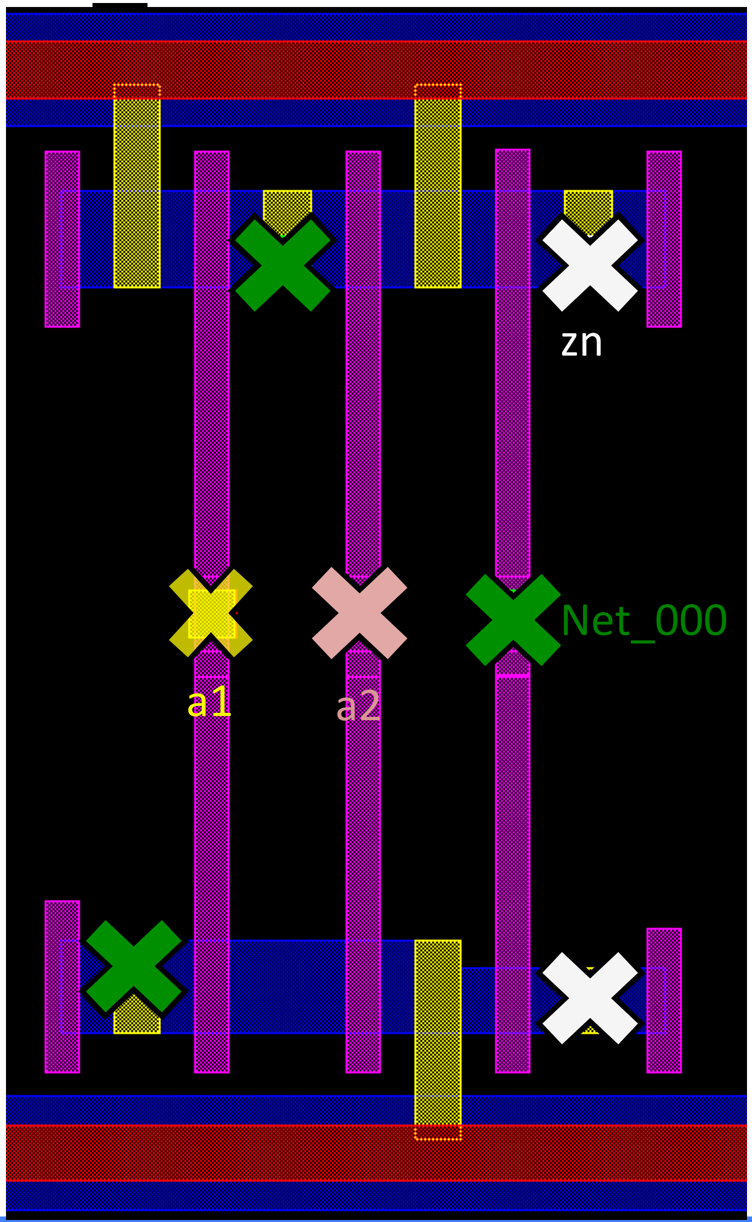
  - OR Wires cross
    - Detection by bitwise ANDing of nodes in the same tile



& = CONFLICT!

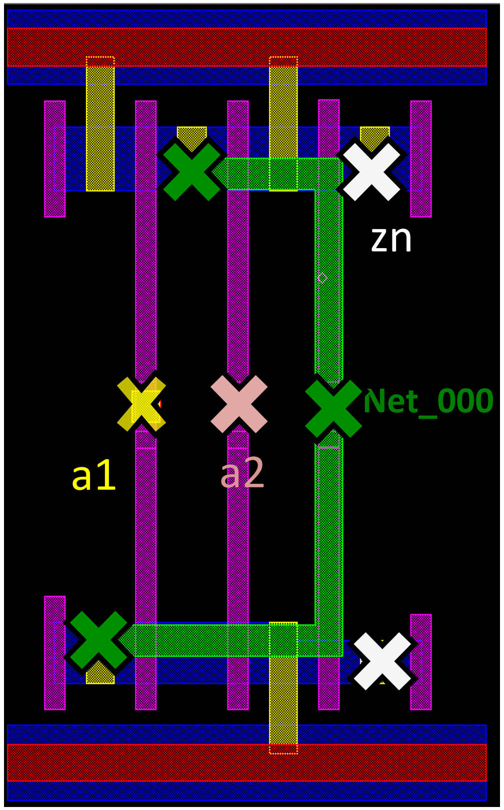
Example: AND2_X1



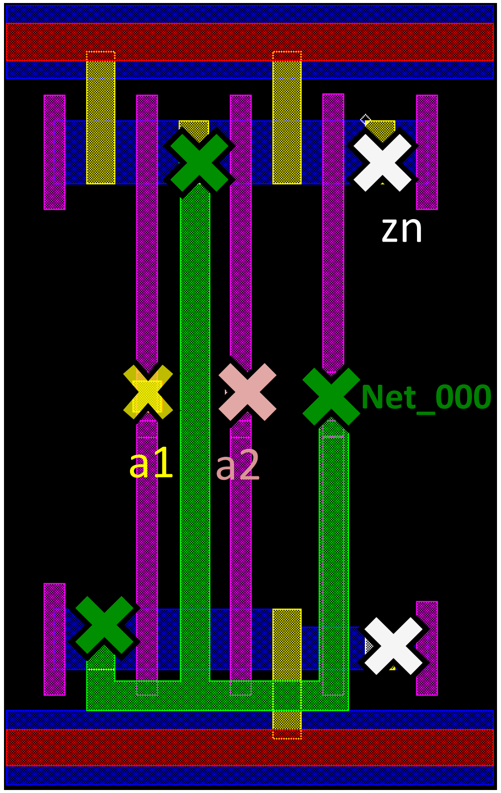


- **4 Nets:**
  - A1 & A2:
    - 2 inputs
    - Each is a single contact net
  - ZN: output
  - Net_000

Example: AND2_X1

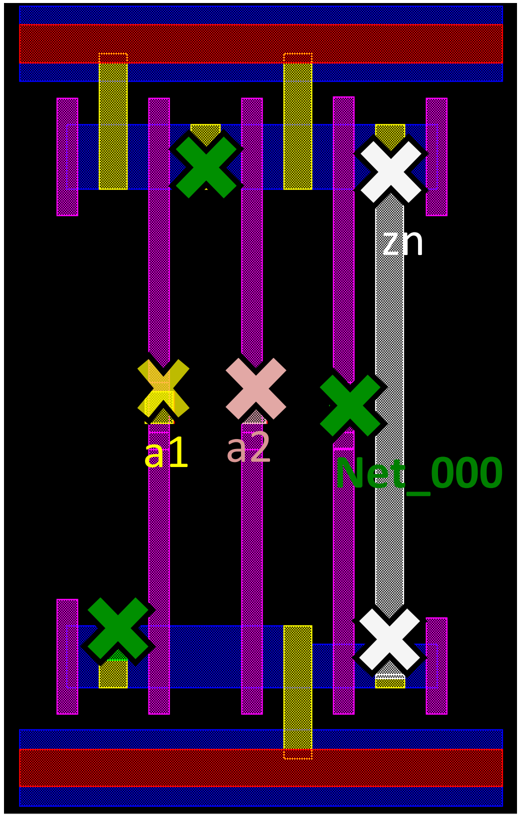


First wiring solution for **net_000**

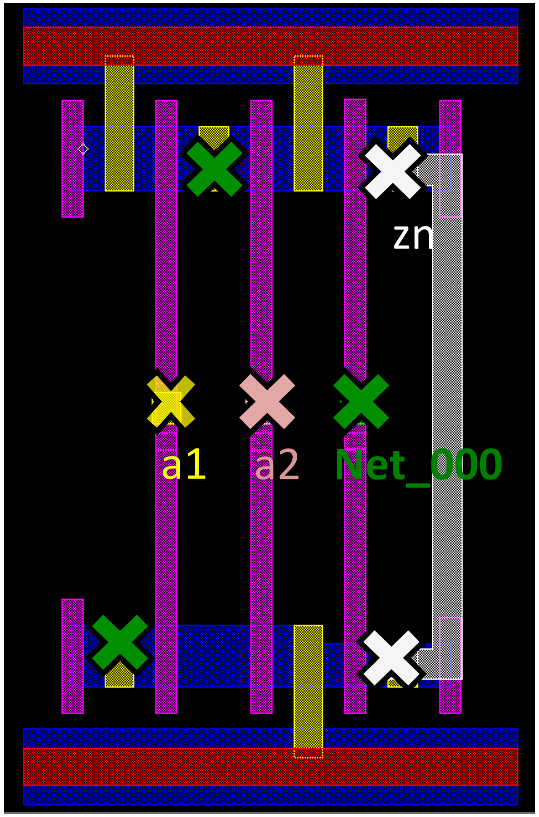


Another wiring solution for **net_000**

Example: AND2_X1



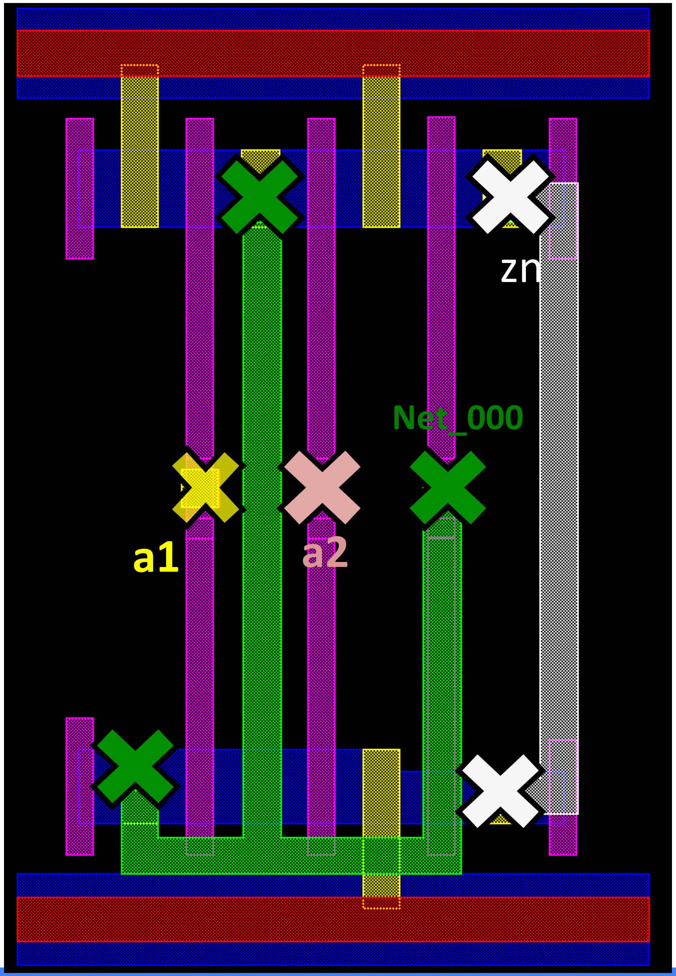
First wiring solution for **zn**

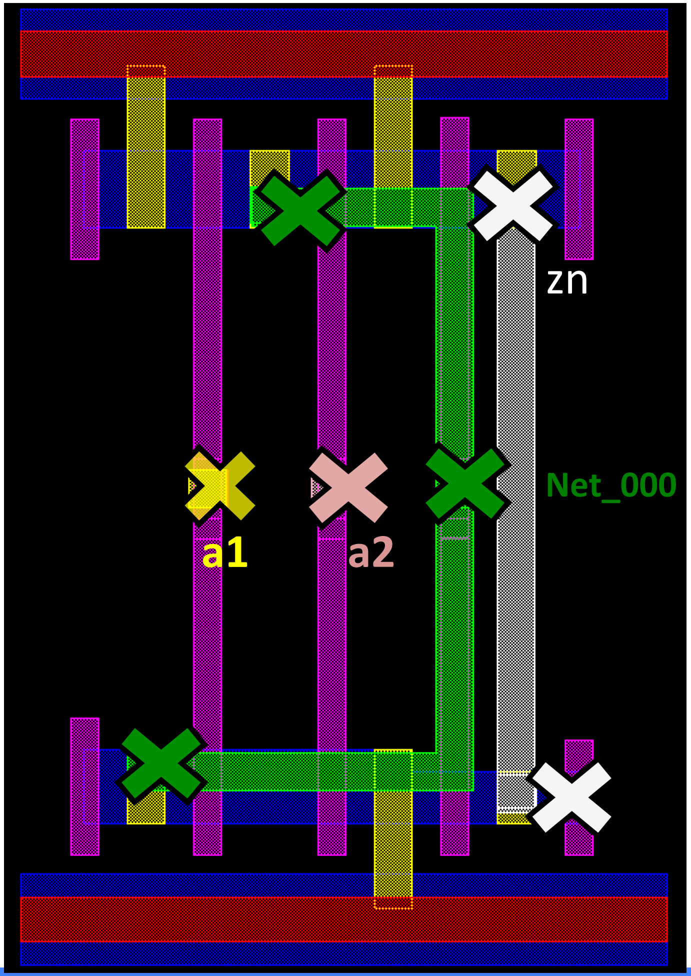


Another wiring solution for **zn**

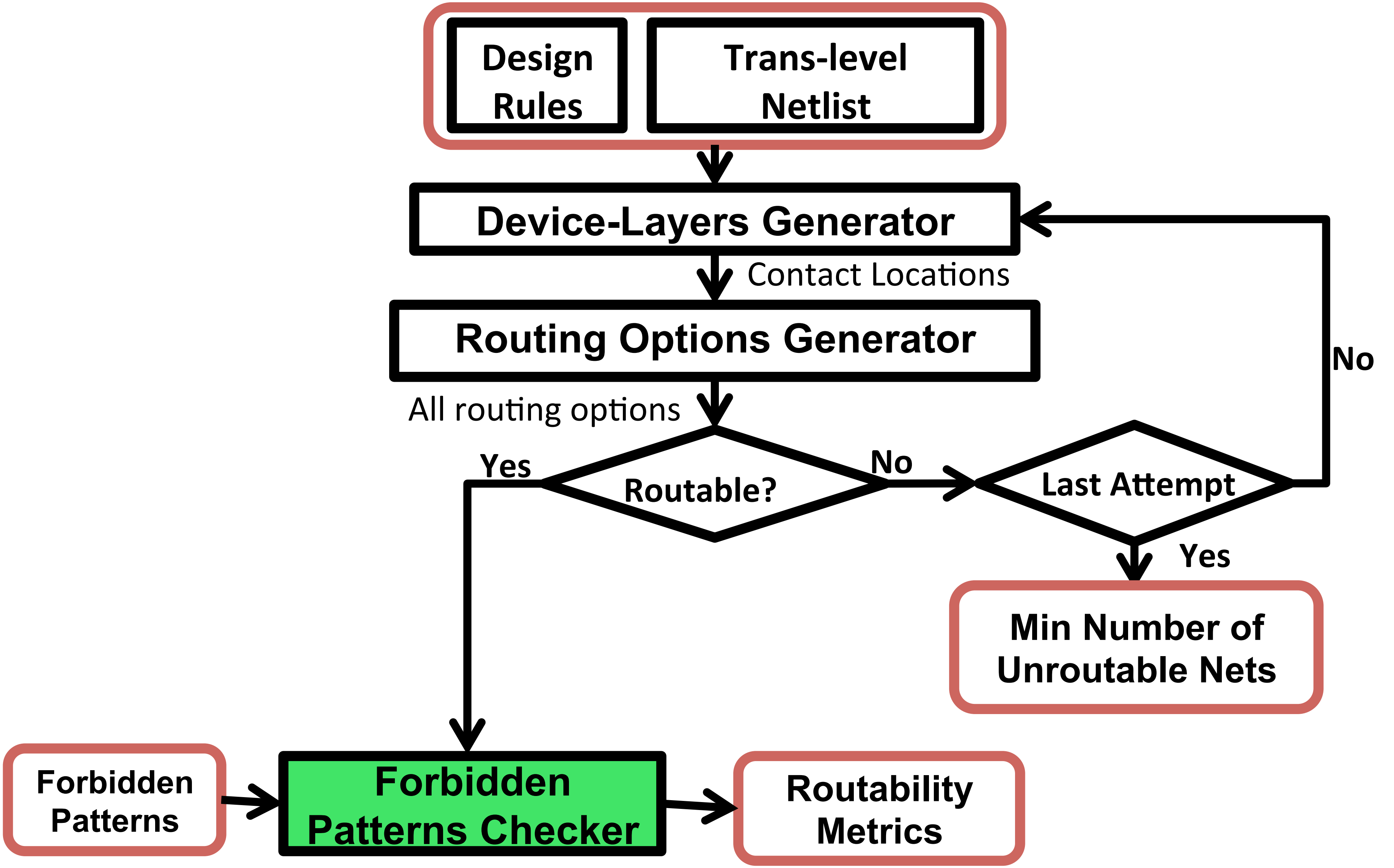
Example: AND2_X1

- Two of the several complete **routing options**





Flow of Pattern-DRE



Design Rules

Trans-level Netlist

Device-Layers Generator

Contact Locations

Routing Options Generator

All routing options

Routable?

Yes

No

Last Attempt

No

Yes

Min Number of Unroutable Nets

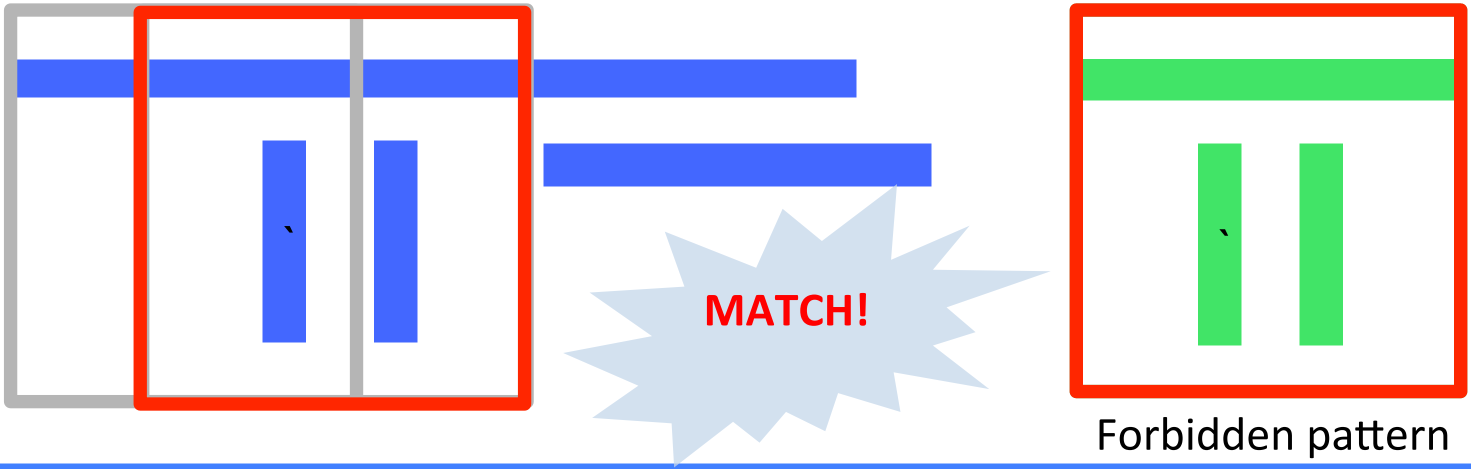
Forbidden Patterns

Forbidden Patterns Checker

Routability Metrics

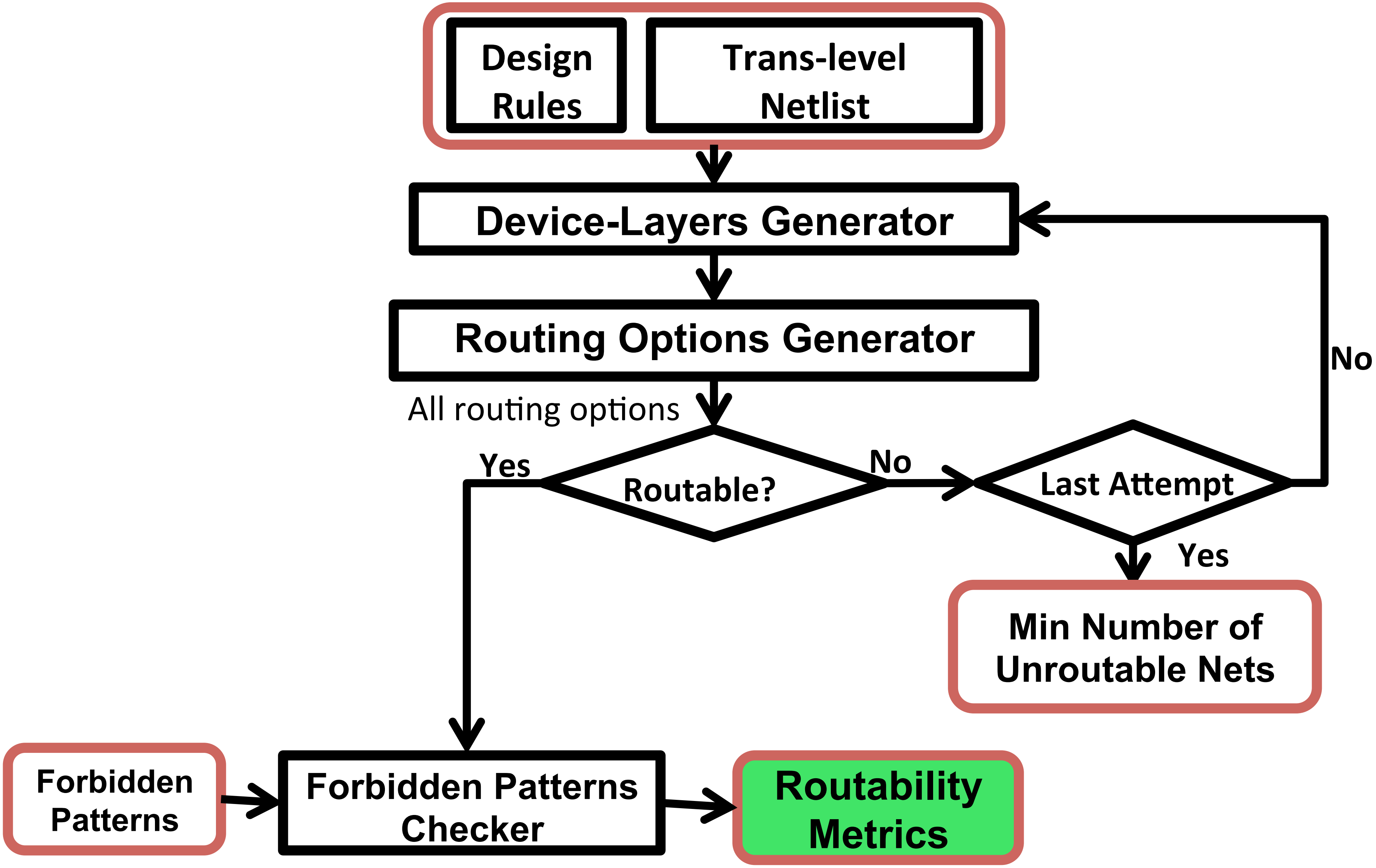
2. Forbidden Patterns Checker

- **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



Forbidden pattern

Flow of Pattern-DRE



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?

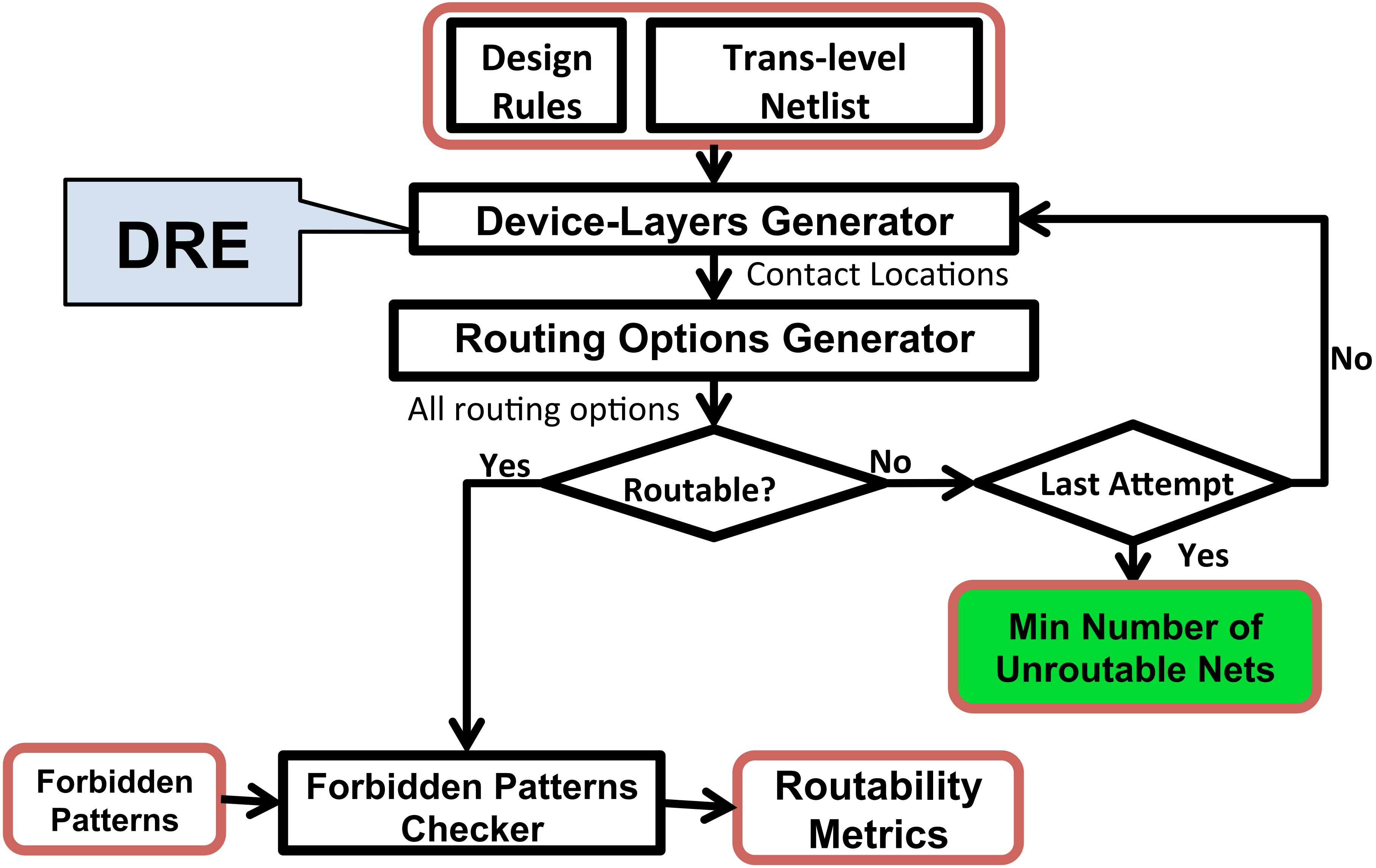
- Given Set A, Set B of forbidden patterns
- Run Pattern-DRE twice
  1. Set A is set of forbidden patterns
  2. Set B is set of forbidden patterns
- If **Set A** has **less routable cells➔ Set A has higher routability impact**
- If same number of routable cells➔ check the **total number of routing options**
  - 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



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

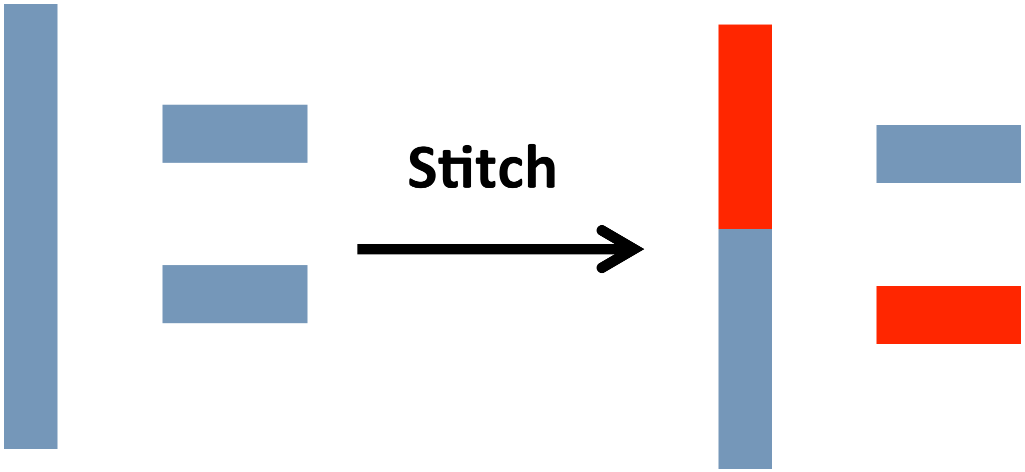
- **Routing Options:** Total number of valid non-forbidden routing options of all cells
- **Routable cells**: Number of cells that have non-zero number of routing options

Experiment #1: SADP vs. LELE

- **<u>Objective</u>**: 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

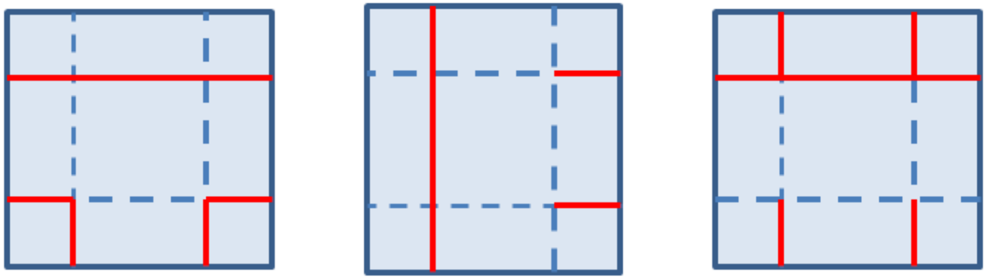
LELE ✔

SADP ✖



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

	Routable Cells	Routing Options	Change in Routing Options
SADP	77	2766	-17%
LELE	78	3338	

- 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

	Routable Cells	Routing Options	Decrease in Routing Options
LELE	**72**	1440	**56.9%**
EUVL	**78**	3338	

- 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**

Diffusion Location	Routable Cells	Routing Options	Decrease in Routing Options
Close to Power rail	**78**	2772	
Close to P/N interface	**74**	861	**6.9%**

Conclusion

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

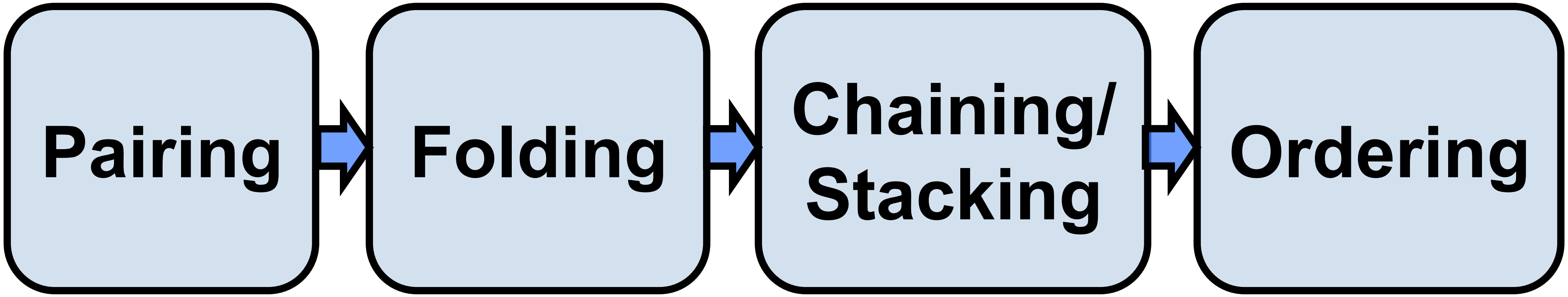
Future Work

- Integrate with a lithography simulator to consider yield-severity of patterns

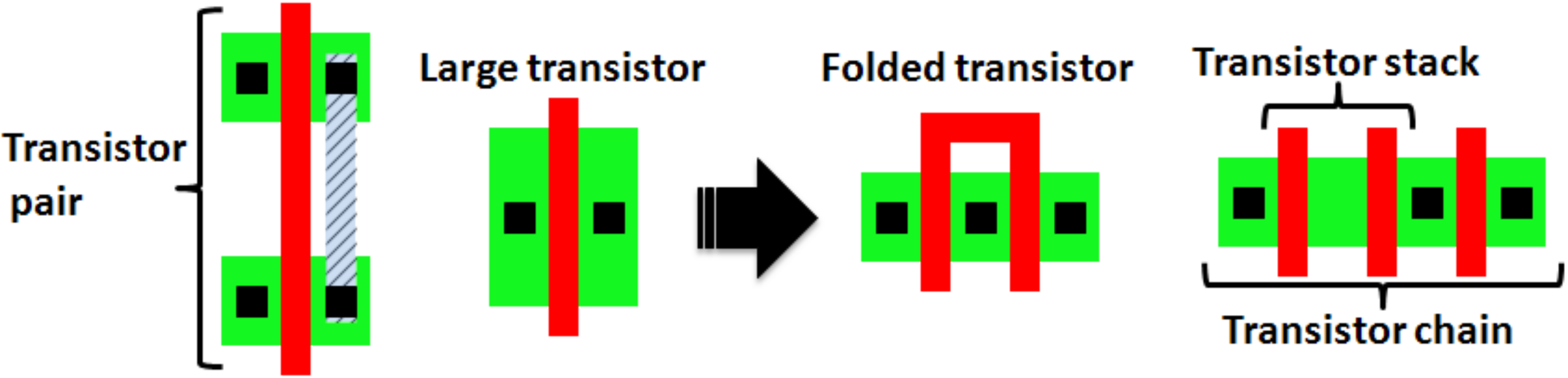
QUESTIONS?

Backup

Device-layers Generator



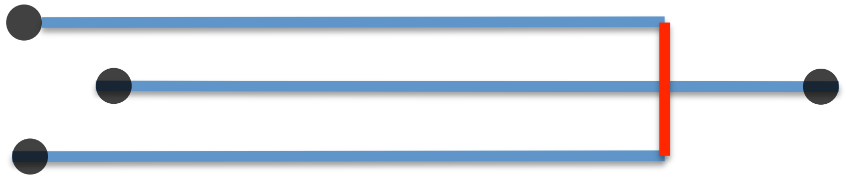
Pairing → Folding → Chaining/ Stacking → Ordering



ICCAD’11, TCAD’12

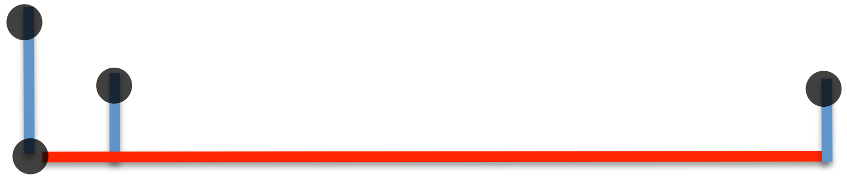
1. Routing Options Generator (cont'd)

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



Vertical Common trunk

Longer WL



Horizontal Common trunk

Horizontal Common trunk

Much shorter WL

- On-track routing