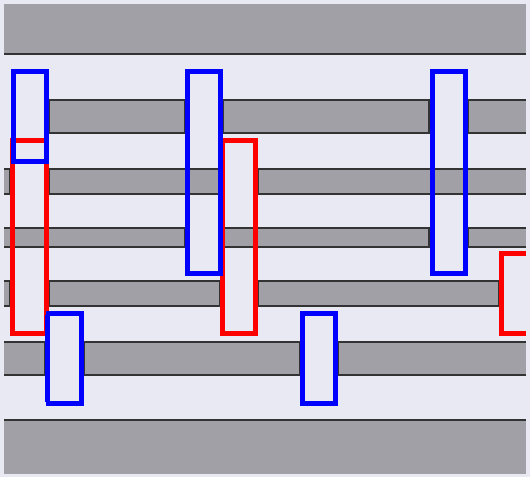
Automatic Cell Layout in the 7nm Era

Pascal Cremer, Stefan Hougardy, Jan Schneider, and Jannik Silvanus

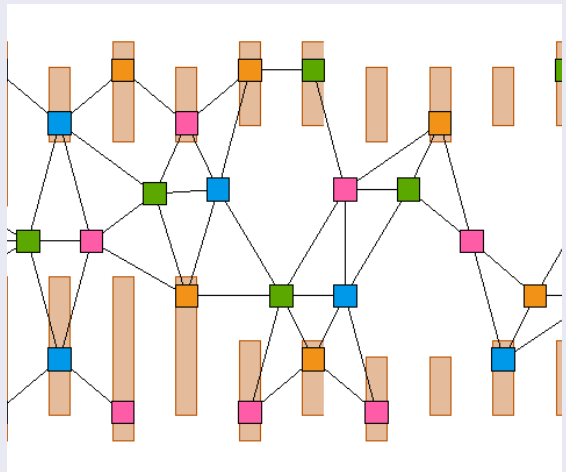
Research Institute for Discrete Mathematics
University of Bonn

March 21, 2017

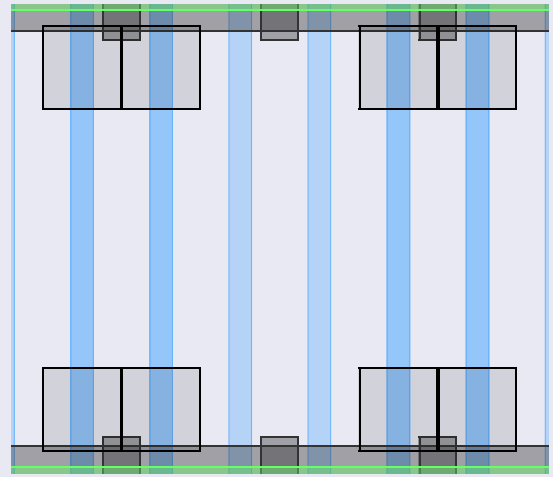
Increasing complexity in 7nm cell design



SADP / SAQP



LELELELE



unroutable placements

Manual cell layout becomes much harder

BonnCell fully automatically builds 7nm physical cell layouts

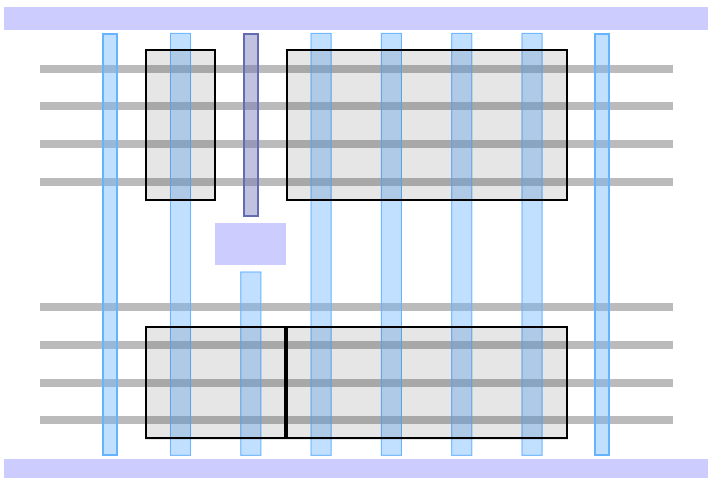
- optimally
- DRC-clean
- DFM-aware

Use cases:

- Interactive prototyping
- Early stage timing analysis
- Highly optimized end stage design

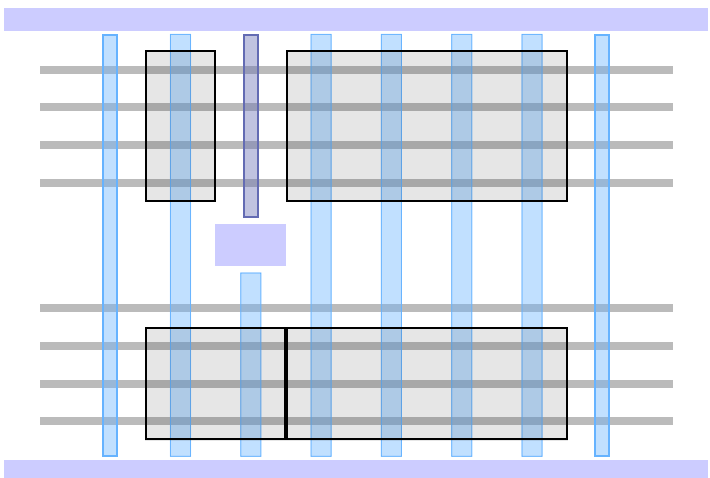
1 Placement

- Branch and bound algorithm
  - check routability
  - minimize area



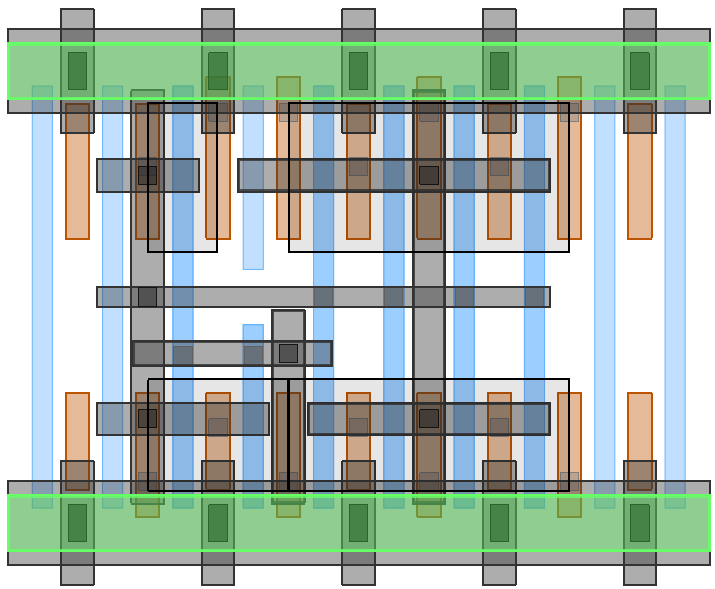
1 Placement

- Branch and bound algorithm
  - check routability
  - minimize area



2 Routing

- MIP routing
  - LVS + DRC clean routing
  - respect DFM constraints



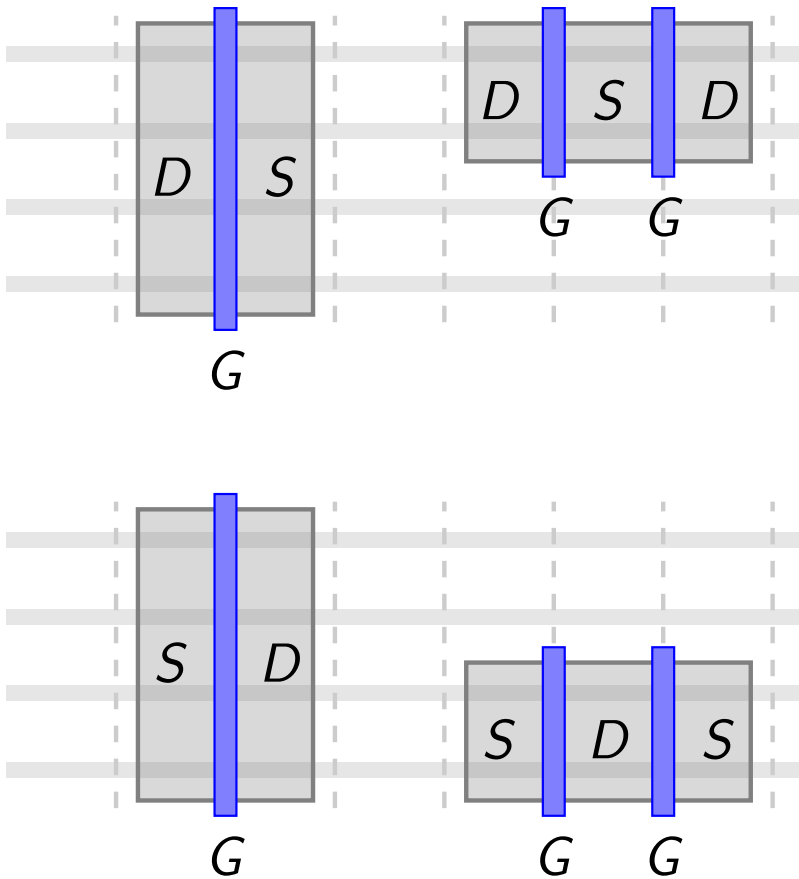
Placement Problem Definition

Given:

- Fets

Output:

- for each fet
  - number of fingers
  - swap status (swapped or not)
  - position



Placement Problem Definition

Given:

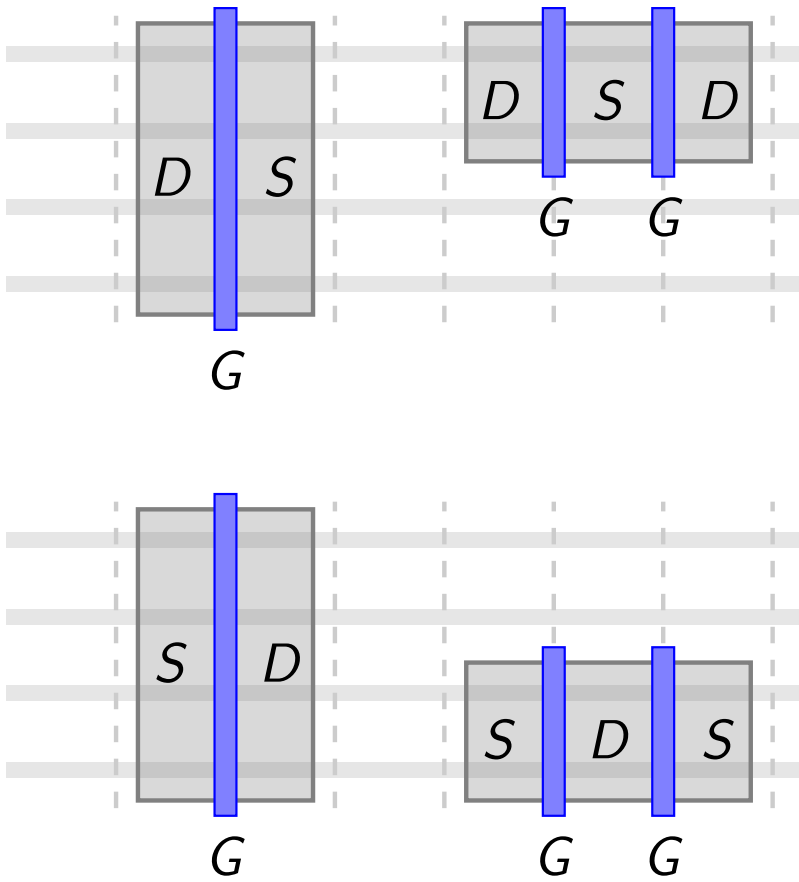
- Fets

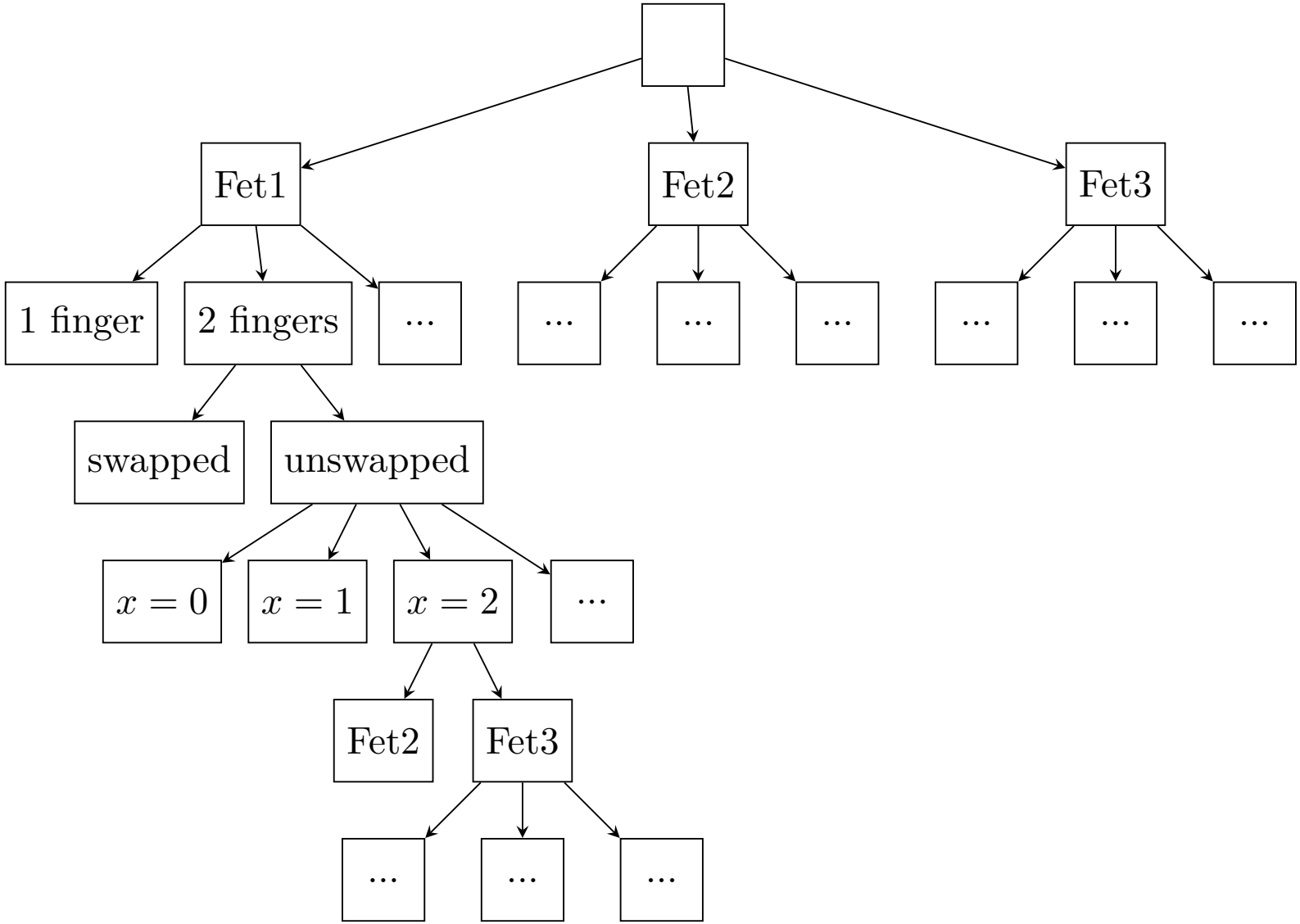
Output:

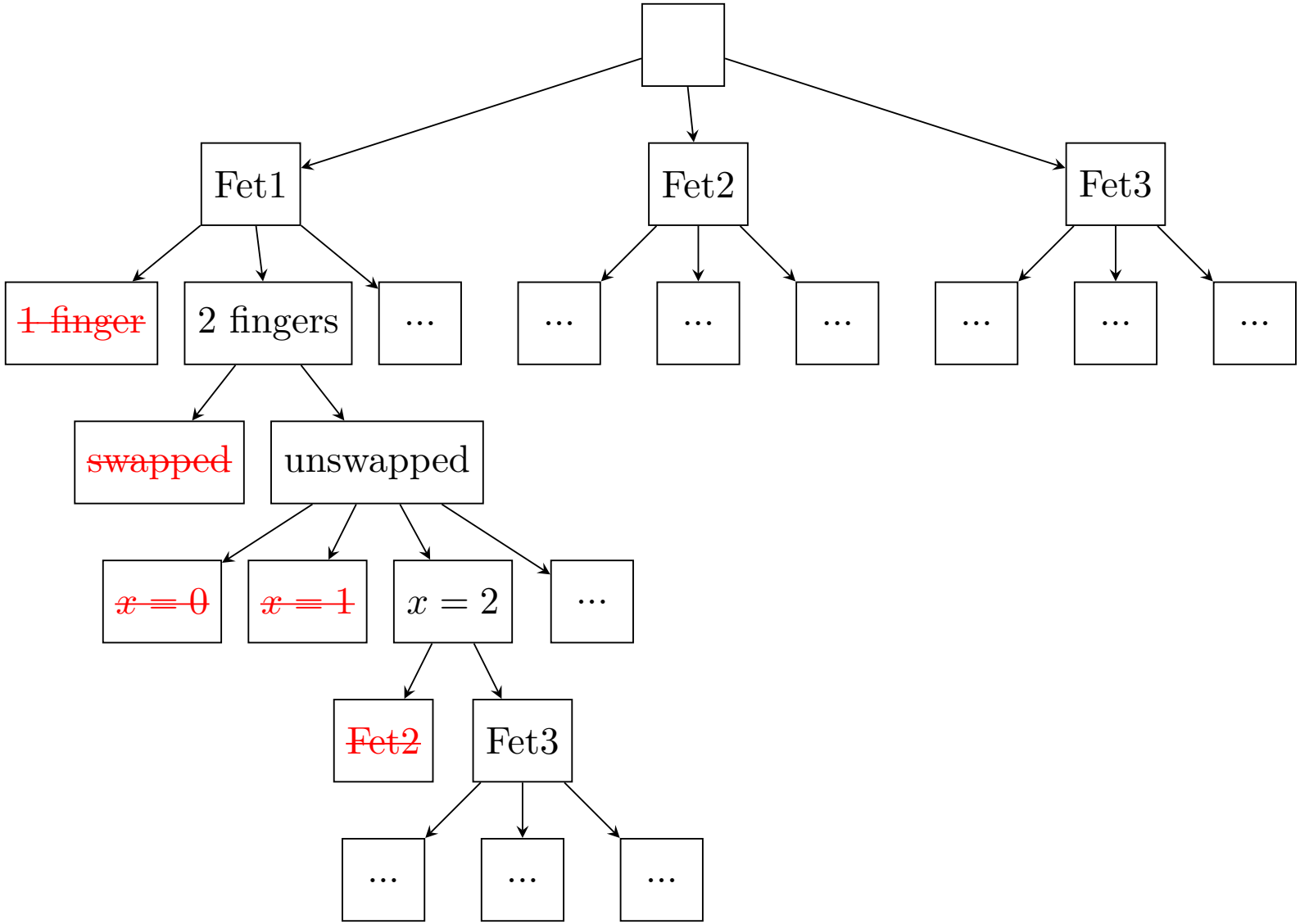
- for each fet
  - number of fingers
  - swap status (swapped or not)
  - position

Target:

- guarantee routability
- minimize cell width
- optimize netlength, timing, ...




Fet1
Fet2
Fet3
1 finger
2 fingers
...
...
...
...
...
...
...
swapped
unswapped
$x = 0$
$x = 1$
$x = 2$
...
Fet2
Fet3
...
...
...



Number of nodes

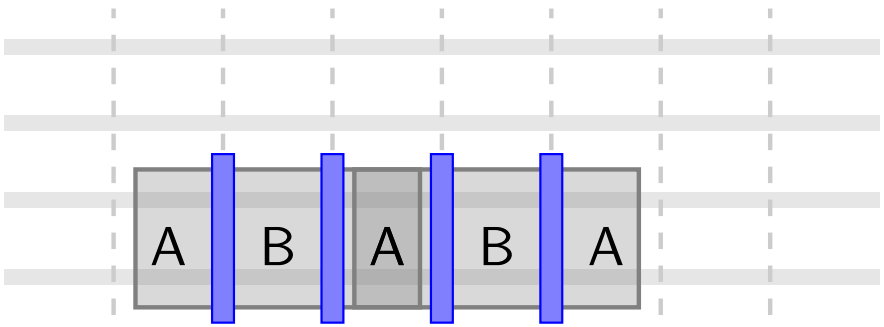
$9.2 \times 10^{14} \rightarrow 7.8 \times 10^{6}$ (9 fets, 15 tracks)

Design Rules

- Two fets can **share** contacts if
  - heights are equal
  - neighboring TS nets are equal
- otherwise they need a gap in between

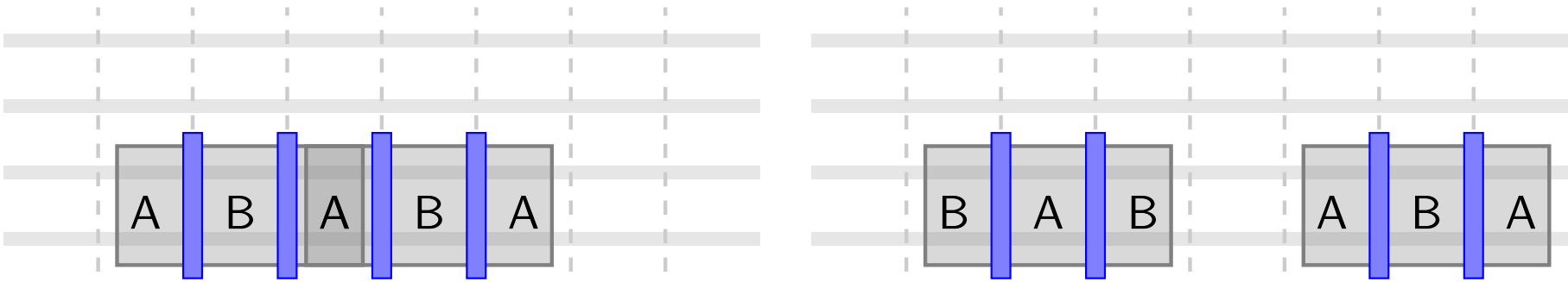
Design Rules

- Two fets can **share** contacts if
  - heights are equal
  - neighboring TS nets are equal
- otherwise they need a gap in between



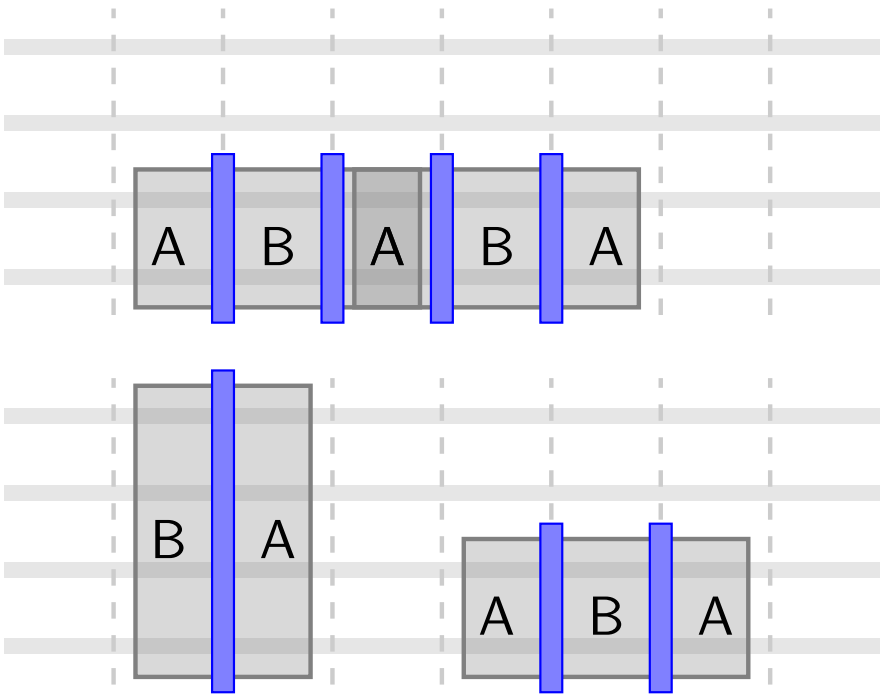
Design Rules

- Two fets can **share** contacts if
  - heights are equal
  - neighboring TS nets are equal
- otherwise they need a gap in between

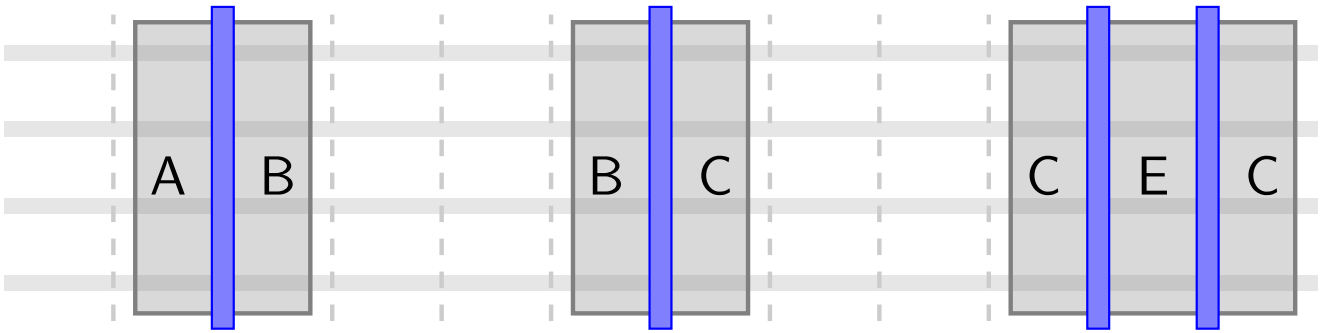


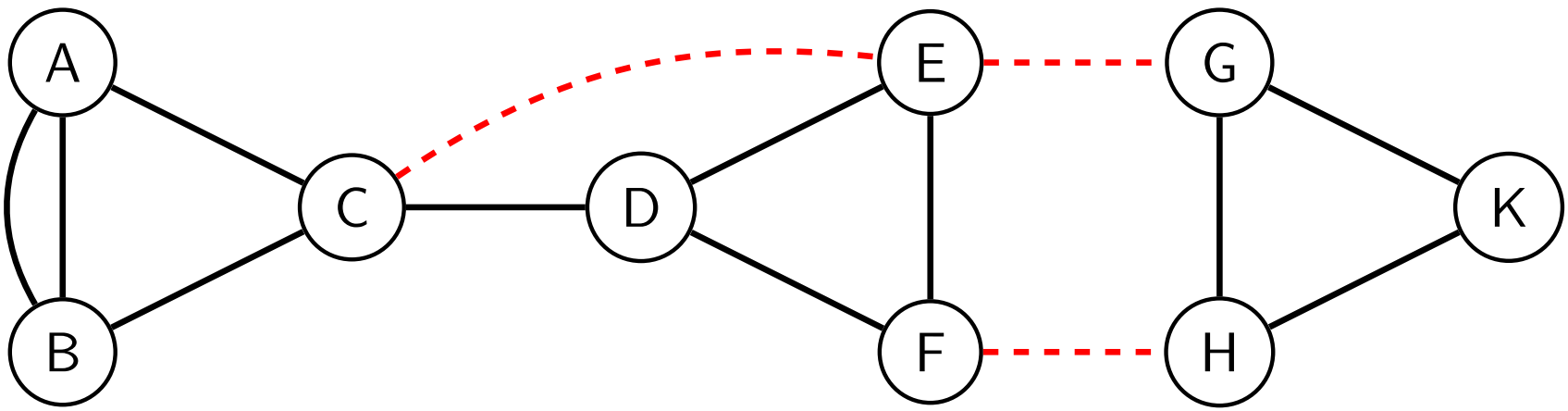
Design Rules

- Two fets can **share** contacts if
  - heights are equal
  - neighboring TS nets are equal
- otherwise they need a gap in between



Graph Formulation

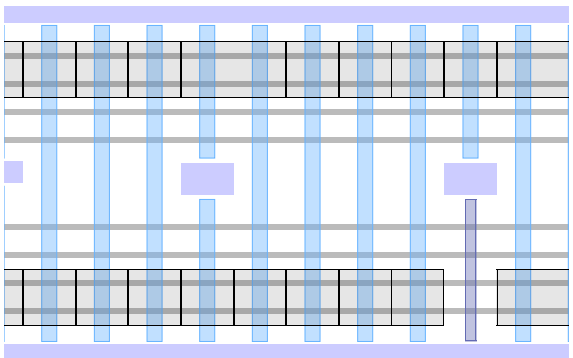




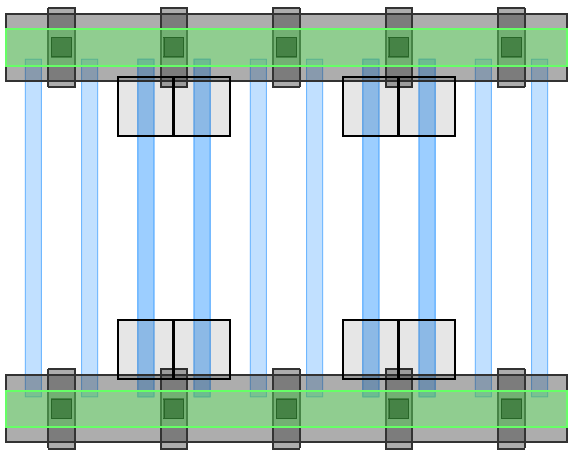
Determine lower bound on placement width by solving

- Minimum Vertex Cover
- Partition into s-t-walks

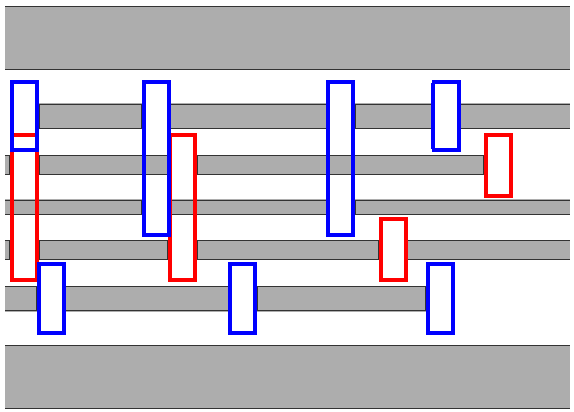
Further Design Rules / Constraints



PC cut shapes



Routability



Mx cut shapes

CT Algorithm

1: **for** $x_1, y_1 \in B^\star \cap [l_1, u_1]$ with $y_1 - x_1 \geq d$ **do**
2: Set $[x_1, y_1]$ as solution of $P_1(x_1, y_1)$
3: **end for**
4: **for** $i = 2, \ldots, n$ **do**
5: **for** $x_i, y_i \in B^\star \cap [l_i, u_i]$ with $y_i - x_i \geq d$ **do**
6: **for** $[x_{i-1}, y_{i-1}]$ s.t. $P_{i-1}(x_{i-1}, y_{i-1})$ has a solution and $[x_{i-1}, y_{i-1}], [x_i, y_i]$ are legal neighbors **do**
7: Set $[x_{i-1}, y_{i-1}], [x_i, y_i]$ as solution of $P_i(x_i, y_i)$
8: **end for**
9: **end for**
10: **end for**
11: Pick legal cut shape on track n and use backtracking to obtain entire solution.

Theorem

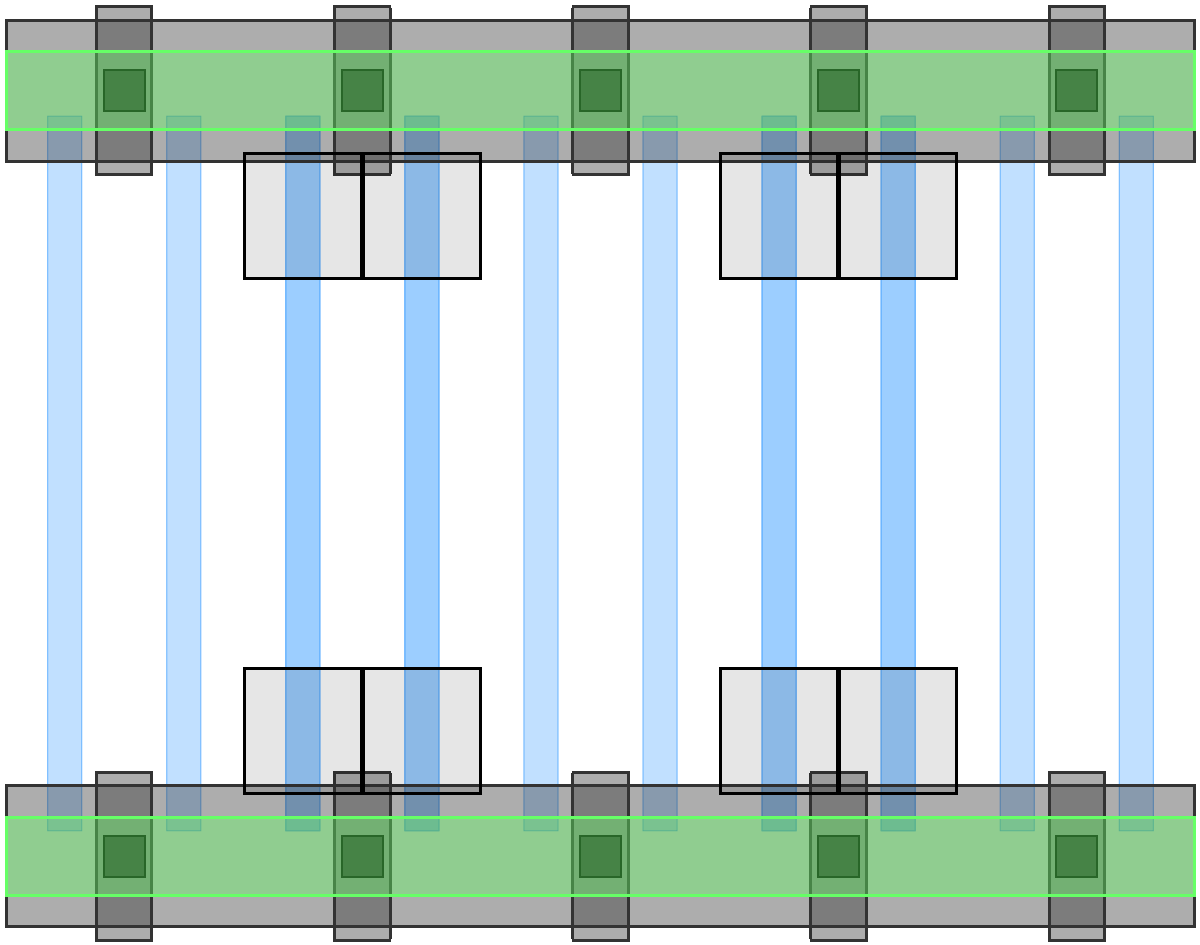
The CT Algorithm solves the PC cut shapes problem in $\mathcal{O}(n^5)$ time, for the number of PC tracks n. In practice it has running time $\mathcal{O}(n)$.

Routing During Placement

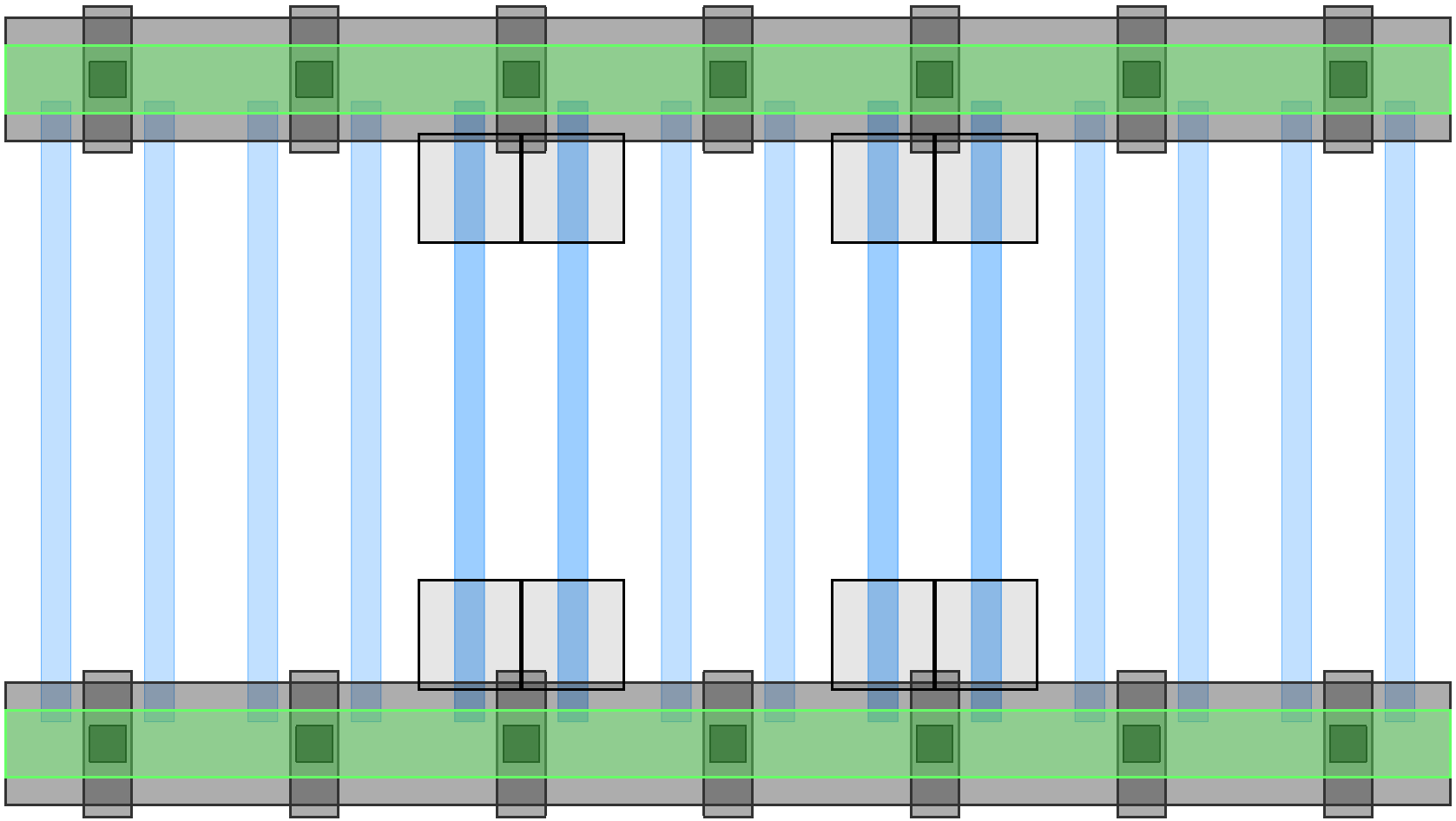
Three modes from a broad spectrum

PC Cut Shapes	Pin Access	Full Routing
• fastest • guarantees legal PC cut shapes	• fast • excludes many unroutable placements	• most expensive • **guarantees routability**

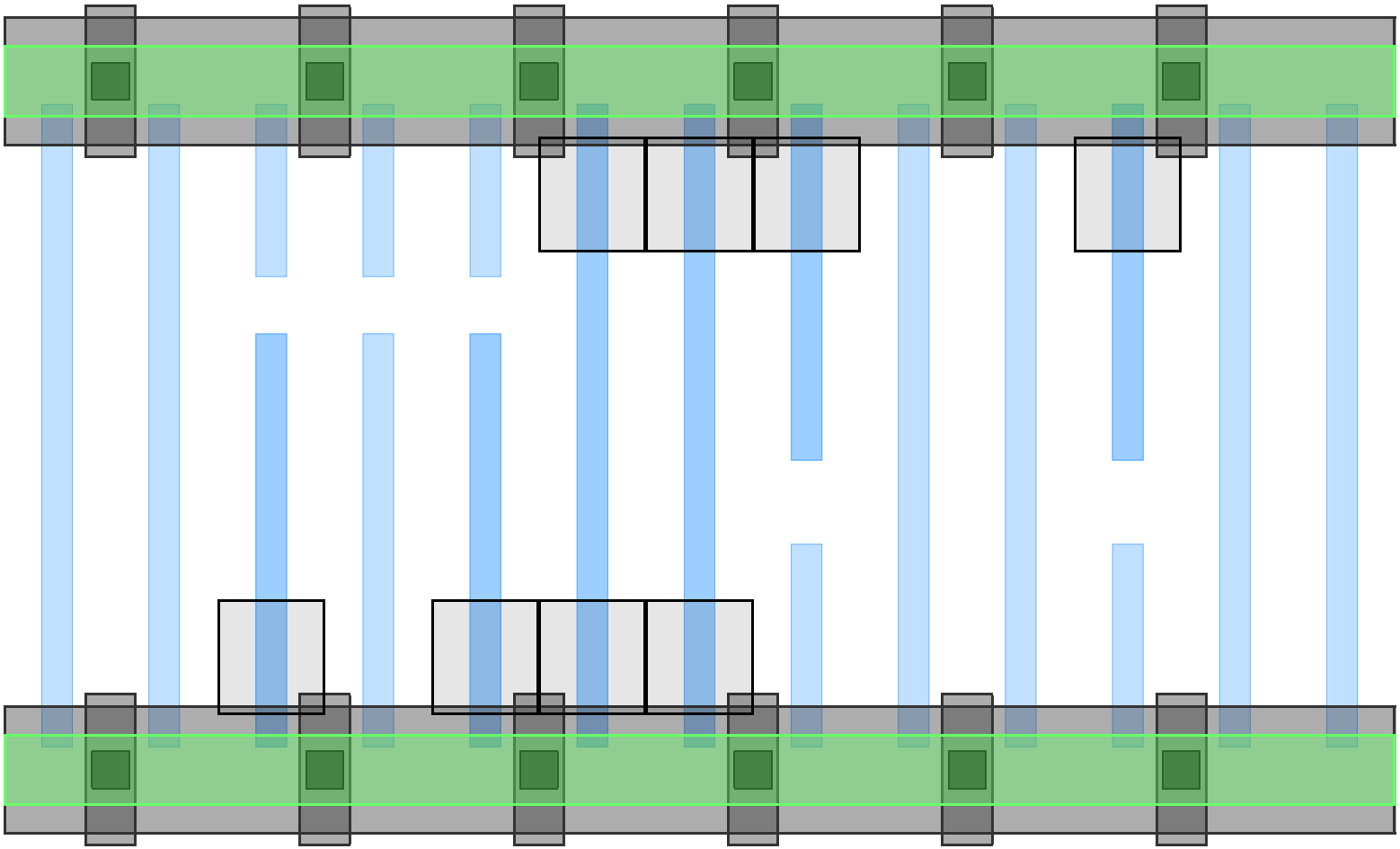
Routing During Placement: Pin Access Mode



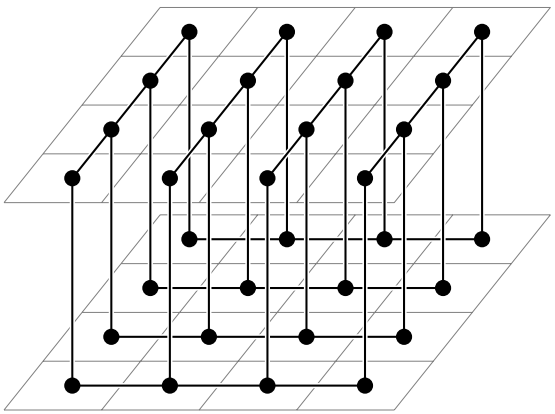
Routing During Placement: Solution Expected by Designer



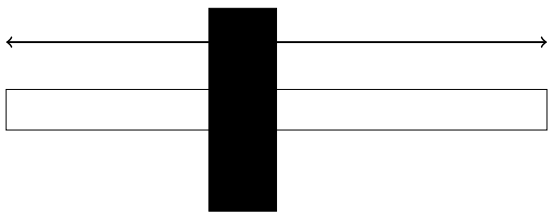
Routing During Placement: Full Routing Mode



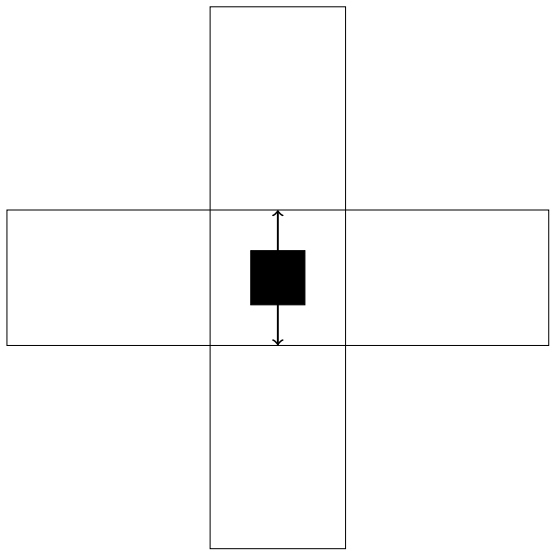
Routing – Features



Grid-based connectivity



Fully flexible metal cut shape positions



Flexible via positions

Routing – MIP Formulation – Connectivity

- MIP - modeled as Steiner tree packing problem in graphs
- State of the art formulations are key to fast running times
  - Unidirected cut relaxation (Integrality gap 2)
  - Bidirected cut relaxation (Worst known example has integrality gap 8/7)
  - **Multicommodity flow relaxation**

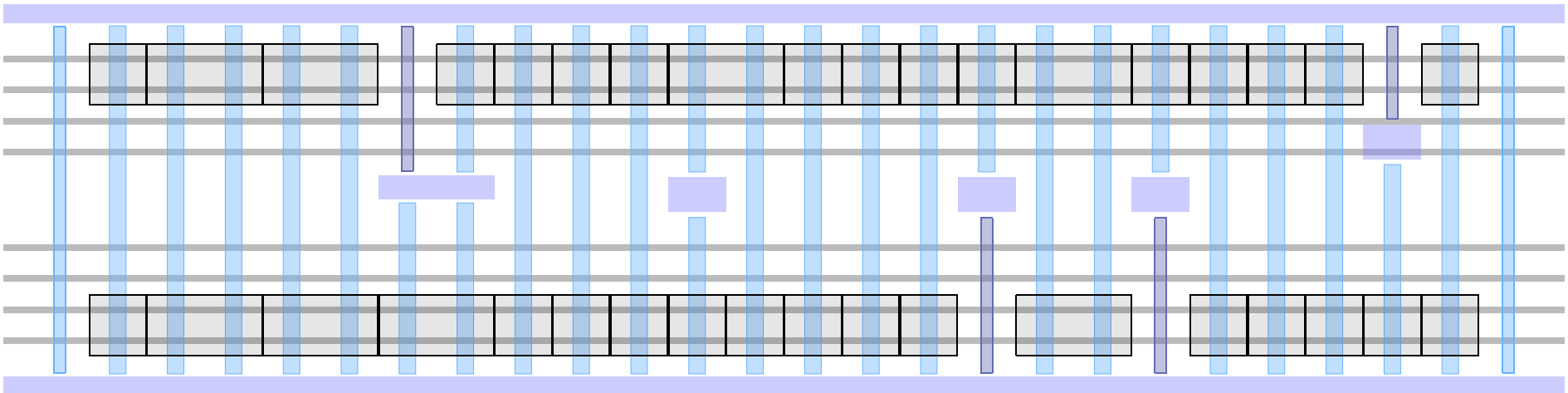
$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e & & \\
\text{s.t.} \quad & x_e = \sum_{k \in \mathcal{N}} x_e^k & & \forall\, e \in E \\
& x_e \in \{0,1\} & & \forall\, e \in E \\
& x_e^k \in \{0,1\} & & \forall\, e \in E, k \in \mathcal{N} \\
& f^t(v) = \begin{cases} 1 & \text{if } i = r_k \\ -1 & \text{if } i = t \\ 0 & \text{else} \end{cases} & & \\
& & & \forall\, v \in V, k \in \mathcal{N}, t \in S_k \\
0 \leq\ & f_{ij}^t \leq \vec{x}_{ij}^k & & \forall\, (i,j) \in A, k \in \mathcal{N}, t \in S_k \\
& \vec{x}_{ij}^k + \vec{x}_{ji}^k \leq x_{\{i,j\}}^k & & \forall\, \{i,j\} \in E, k \in \mathcal{N}
\end{aligned}
$$

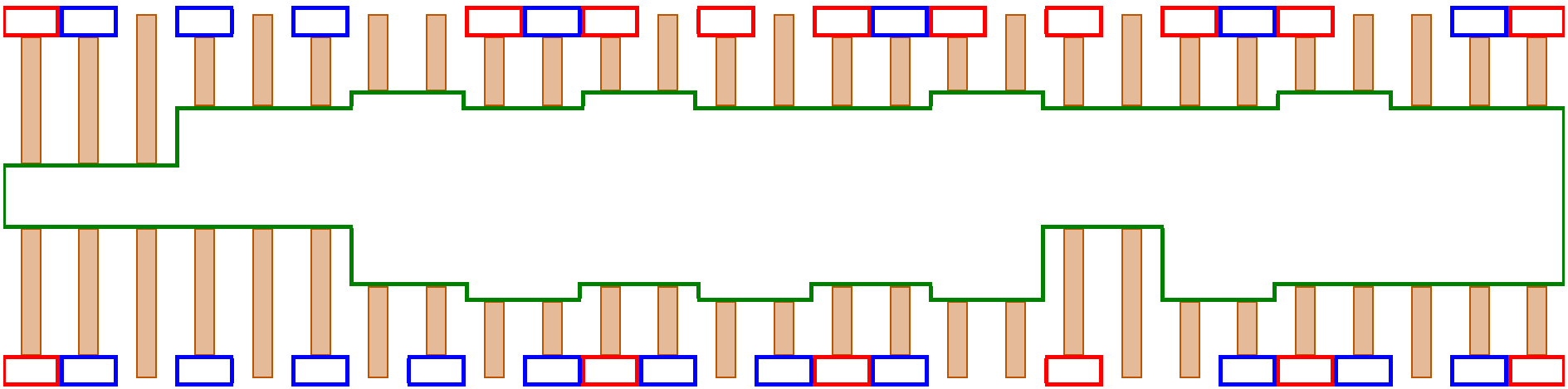
Routing – Design Rules

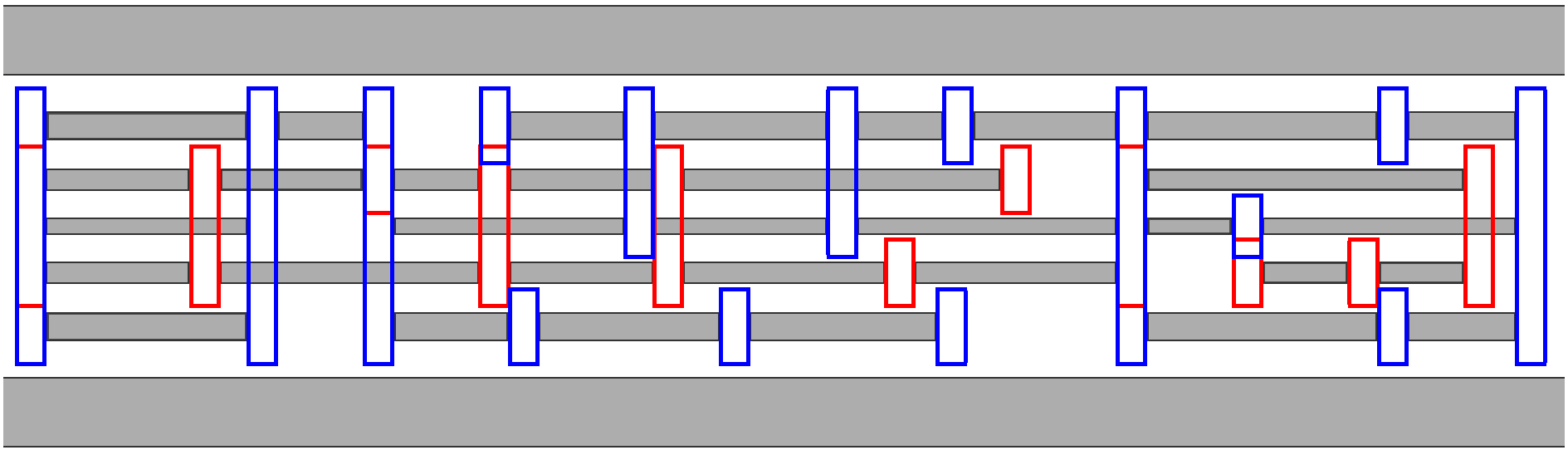
Exact representation of **all** Design Rules (DRC + DFM)

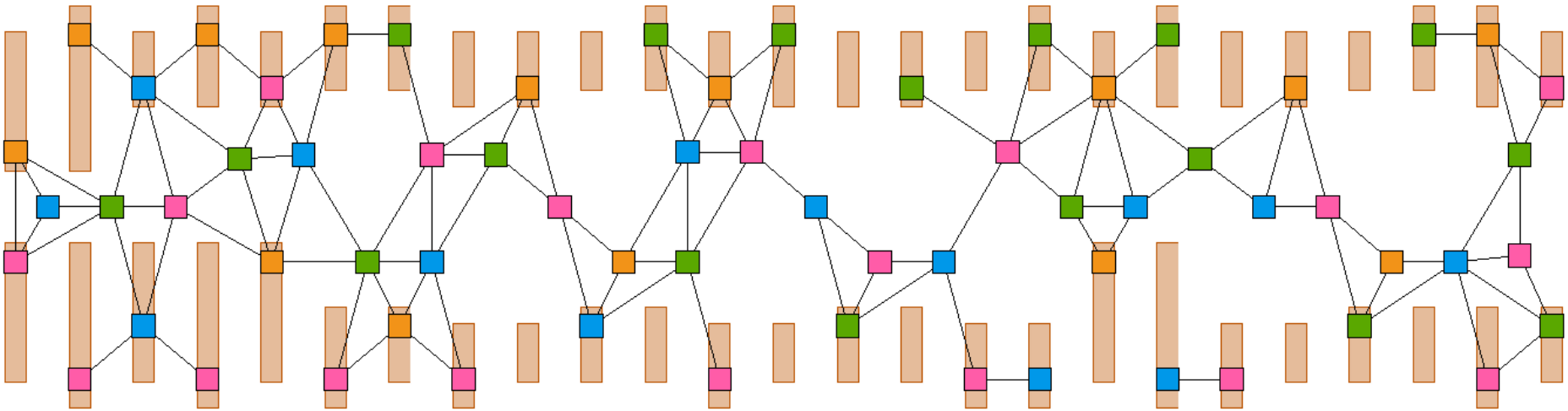
- Cut shapes – cut shape spacing
- Via metal overhangs
- Metal min area
- Via coloring
- Via – via spacing with flexible via positions
- Many more ...

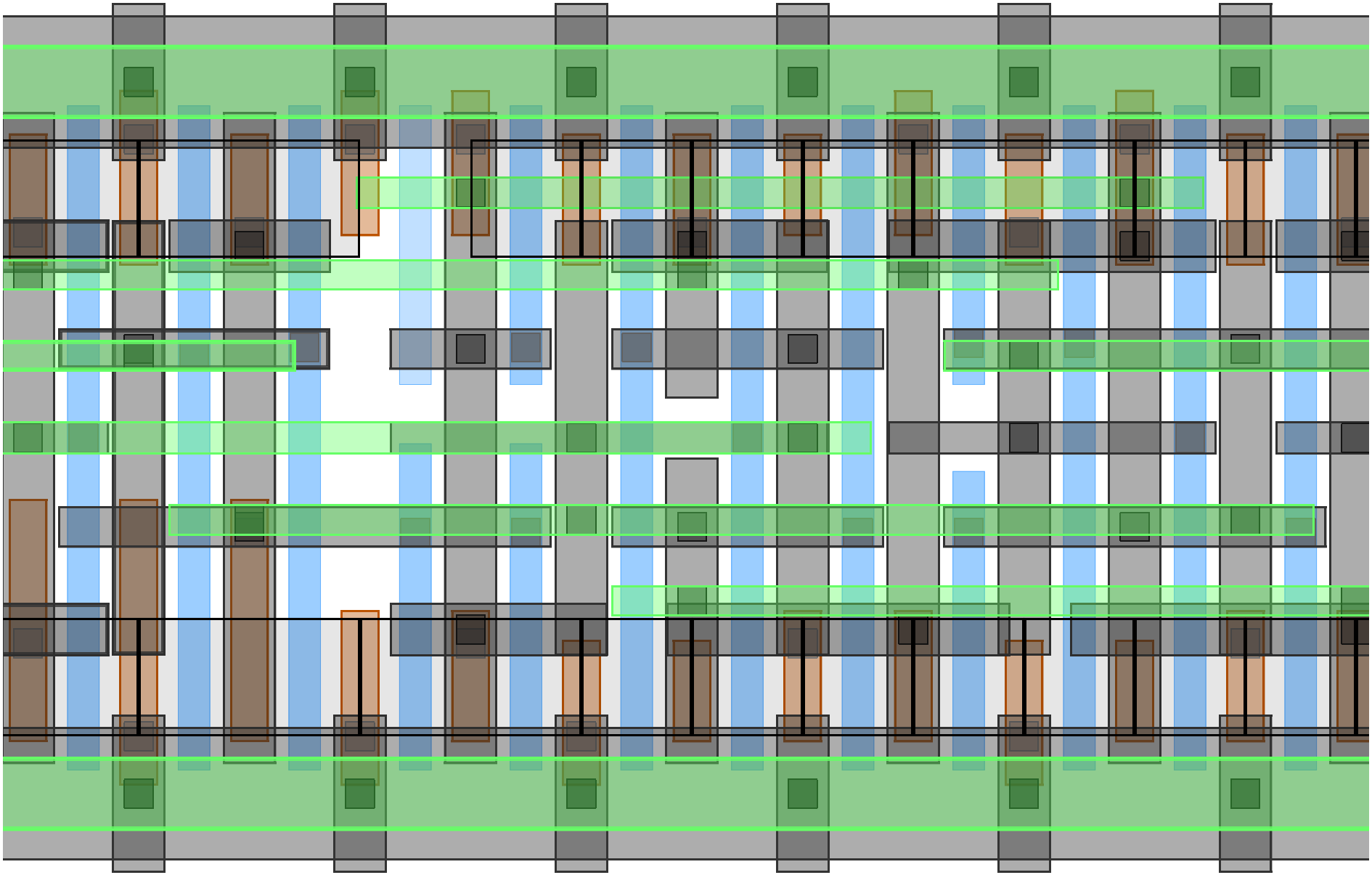
Full Optimization of **netlength**











Standard logic

- 2 – 14 fets
- 5 – 16 nets
- 4 – 12 CPP

14nm comparison:

- BonnCell improves area for 43% of all library cells

Latches

- 28 – 36 fets
- 21 – 28 nets
- 22 – 32 CPP
- highly complex
- used many times on chip
- Manual layout work: weeks

Standard logic

- 2 – 14 fets
- 5 – 16 nets
- 4 – 12 CPP

14nm comparison:

- BonnCell improves area for 43% of all library cells

Latches

- 28 – 36 fets
- 21 – 28 nets
- 22 – 32 CPP
- highly complex
- used many times on chip
- Manual layout work: weeks

BonnCell

Minimal area **Placement**

LVS, DRC, and DFM clean **Routing**

Standard logic $\leq$ **6min** Latches $\leq$ **19h**

BonnCell fully automatically builds 7nm physical cell layouts

- optimally
- DRC-clean
- DFM-aware

Use cases:

- Interactive prototyping
- Early stage timing analysis
- Highly optimized end stage design

Thank you!