Pin Accessibility-Driven Detailed Placement Refinement

Yixiao Ding @ Cadence Design System
Chris Chu @ Iowa State University
Wai-Kei Mak @ National Tsing Hua University
Outline

• Introduction
  › Why pin access is a critical problem
  › Previous works and our motivation
• Overview of our solution
  › Pin accessibility-driven detailed placement (DP) refinement
  › Our contributions
• Problem formulation
• Background knowledge
  › Assumptions
  › Pin access region
  › Pin access penalty
• Proposed solution
• Experimental results
A critical pin access problem (1/2)

- Unidirectional routing trend in advanced nodes
  - Metal-1 pins can be easily blocked by straight metal-2 wires
  - Fierce routing resource competition on metal-2
A critical pin access problem (2/2)

- More restricted routing design rules in advanced nodes
  - e.g., more space between vias
  - e.g., metal layer patterns are compliant to SADP design rules
Improve pin access in different design stages

- routability-driven
- local congestion-aware
- Pin access planning

Physical design flow:
- Global placement
- Detailed placement
- Global routing
- Detailed routing
Selected previous works (1/3)

- Routability-driven global placement
    - Cell spreading in congested region. too rough
- Local congestion-aware detailed placement
    - Identify hard-to-route cell based on pin area and resolution. not exact
- Local congestion and pin access-aware global routing
  - C. Alpert et al, “Consideration of local routing and pin access during VLSI global routig”, US Patent’ 13
    - Consider pin count, relative location, and Steiner tree length. limited
- Pin access planning in detailed routing (DR)
  - Next two slides
Pin access planning in DR (1/2)


![Diagram showing pin access planning examples]

- A bad planning
- A wise planning
pin access planning in DR (2/2)

- It is not always effective, especially in areas with high pin density.

- Motivation: we want to resolve pin access issues here!

```
SCa  SCb
```

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Our proposed solution

physical design flow

- global placement
- detailed placement
- global routing
- detailed routing

PA-driven DP refinement

Refinement techniques:
Phase 1: cell flipping & swap
Phase 2: cell shifting

to improve pin access/routability
DP refinement techniques (1/2)

- Consider pin access in DP, when cell movement is allowed
- Cell shifting, adjacent cell swap, and cell flipping

shift SC\textsubscript{b} to right
DP refinement techniques (2/2)

flip SC\textsubscript{b}

swap SC\textsubscript{a} and SC\textsubscript{b}
Our contributions

• It is the first work to directly consider pin access issue in detailed placement (DP) stage

• An accurate model is proposed to capture pin access scenario in detailed routing. A cost function is presented to guide DP refinement to improve pin access

• Our DP refinement techniques are limited to cell flipping, adjacent cell swap, and cell shifting. Our proposed solution is dynamic programming and linear programming-based.
  ‧ Respect the given placement solution
  ‧ Guarantee good solution quality with fast runtime

• Experimental results demonstrates the effectiveness of our proposed pin access-driven DP refinement
Problem formulation

- **Given**
  - A legalized placement
- **We try to refine the placement by cell shifting, cell flipping, and adjacent cell swap.**
- **Objective**
  - Pin accessibility / routability is improved in DR stage
  - Placement perturbation should be minimized
  - The overheads of WL, via count in DR solution should be small or unchanged
- **Constraint**
  - Refined placement is legal
Assumptions

• Each metal layer has a preferred routing direction.
  › Metal1 is unroutable. metal2 horizontal, metal3 vertical…
• Standard cell (SC) ’s pins are rectilinear polygons on metal1. Each pin may span several metal-2 tracks.
  › The tapping point (TP) of a pin is defined as the overlap of metal-2 track and the pin shape

Pin access is to select a TP as a via location to connect metal-1 pin and metal-2 wire segment such that the metal-2 wire segment can be extended toward conn. dir. until the other connected pin.
Pin access region (PAR)

• Given a 3-pin net \{A, B, C\}, a PAR for is defined for each connection of each pin
  ‣ Same-row connection AB
  ‣ Different-row connection BC
Pin access penalty (PAP) (1/3) - penalty function

- Objects (e.g., metal-2 block, wire segment) in PAR are penalized.

- Penalty function $f_w(dist)$

![Graph showing different-row connection]

- $f_w(dist)$ for same-row connection
- $f_w(dist)$ for different-row connection

$1$ for minimum width $min_w$ of metal-2 wire
Pin access penalty (PAP) (2/3)

- For each connection, PAP reflects its pin accessibility
- Given a connection, a conflicting block (CB) is a block obstruct its PAR
- Given a connection, a conflicting connection (CC) is a connection with a PAR intersects with its PAR.
- PAP of a connection is computed by accumulating the penalty cost due to all the CBs and CCs.

\[
PAP_{AA'} = \sum_{\text{block} \in CB} PAP_{AA'}^{\text{block}} + \sum_{\text{conn} \in CC} PAP_{AA'}^{\text{conn}}
\]

- PAP with a CB or CC = probability x \(f_w(\text{dist})\)
  - Four different scenarios, and equations are different in different scenarios
Pin access penalty (PAP) (3/3)
- four scenarios

(1)

(2)

(3)

(4)
Quantify the pin accessibility of a DP

- For each connection, pin access penalty (PAP)

\[ PAP_{AA'} = \sum_{\text{block} \in CB} PAP_{block}^{\text{block}} + \sum_{\text{conn} \in CC} PAP_{AA'}^{\text{conn}} \]

- For each cell c, cell pin access penalty (CPAP)

\[ CPAP_c = \sum_{A \in \text{Pin}_c} \sum_{AA' \in \text{Conn}_A} PAP_{AA'} \]

- For a DP, total cell pin access penalty (TCPAP)

\[ TCPAP = \sum_{c \in \text{All\_Cells}} CPAP_c \]

- TCPAP reflects pin accessibility of a DP, and should be minimized
Two-phase PA-driven DP refinement

- Phase 1: cell flipping & cell swap
  - dynamic programming row by row
- Phase 2: cell shifting
  - linear programming
Refinement phase 1 (1/3)

- Given a row of placement, we try to minimize \( \sum_{c \in C_{row}} CPAP_c \)

- Dynamic programming
  - \( \text{Sol}_k \) contains optimal refined prefix placement with \( k \) cells
  - Base cases \( \text{Sol}_1 \)

![Diagram of row placement with cells and summation formula]

[Diagram showing placement and summation formula]
Refinement phase 1 (2/3)

- Recursive formula
  - 6 kinds of solk should be kept in dynamic program to obtain soln, each kind is determined by the last cell placement in solk
  - Given a solk, solk+1 is obtained by finding min. total CPAP for cells in solk+1
  - An example to obtain sol4 from sol3

```
sol3
... 3 ... 
... 3 ... 
... 3 ... 
... 2 ... 
... 2 ... 
... 2 ... 
... 4 ... 
... 4 ... 

sol4
...... 3 4 ...... 
...... 3 4 ...... 
...... 3 4 ...... 
...... 2 4 ...... 
...... 2 4 ...... 
...... 2 4 ...... 
```

22
Refinement phase 1 (3/3)

- Opt is obtained by finding min. total CPAP among 4 kinds of soln

\[
\begin{array}{c}
\ldots \ldots \\
\ldots \ldots \\
\ldots \ldots \\
\ldots \ldots \\
\end{array}
\]

\[
\begin{array}{c}
\text{n} \\
\text{n} \\
n-1 \\
n-1 \\
\end{array}
\]
Objective: minimize TCPAP
  › Linear approximation on penalty function
Continuous variable denotes the x-location of each cell’s lowerleft corner
Δ controls the threshold of max cell shifting distance
Linear constraints to ensure cells are not overlapped and out of LL & RR
Experimental set-up

• PA-driven DP refinement is implemented by C++.
• Experiments run on 2.4 GHz Intel Core i5 and 8GB memory.
• Gurobi 6.0 is called to solve linear program in phase 2.
• Original benchmarks are from [1].
• SADP-aware detailed router in [2] is called to route refined DP.
• Two sets of experimental results are demonstrated:
  › PA-DP refinement
  › Detailed routing on refined placement
PA-driven DP refinement (1/2)

- Benchmark statistics

<table>
<thead>
<tr>
<th></th>
<th>ecc</th>
<th>efc</th>
<th>ctl</th>
<th>alu</th>
<th>div</th>
<th>top</th>
</tr>
</thead>
<tbody>
<tr>
<td>#cells</td>
<td>1302</td>
<td>1197</td>
<td>1715</td>
<td>1802</td>
<td>3260</td>
<td>12576</td>
</tr>
</tbody>
</table>

Nor. TCPAP

- P1 ave. = 0.85
- P2 ave. = 0.82

CPU (s)

- Phase1 & 2 CPU
- Phase1 CPU
PA-driven DP refinement (2/2)

Ave. displacement (pitch)

P2 ave. = 6.73
P1 ave. = 5.04

Pct. flipped cell

P1 ave. = 0.33
Detailed routing results (1/2)

Given DP

- P1 ave. = 0.77
- P2 ave. = 0.67

DP after p1

- P1 ave. = 1.14
- P2 ave. = 1.09

DP after p2

Nor. UnroutableNets

- ecc
- efc
- ctl
- alu
- div
- top

Nor. WL

- ecc
- efc
- ctl
- alu
- div
- top
Detailed routing results (2/2)

P1 ave. = 1.02
P2 ave. = 1.05

P1 ave. = 1.06
P2 ave. = 1.15
Thank you!

Q & A