Seeing the Forest and the Trees: Steiner Wirelength Optimization in Placement

Jarrod A. Roy, James F. Lu and Igor L. Markov
University of Michigan Ann Arbor
April 10, 2006
Motivation

- Place-and-route
  - Single step for designers?
  - Implemented as separate point tools
  - Very little interaction/communication
  - Use different optimization objectives
- Our goal: reduce the gap between placement and routing
  - HPWL is the wrong objective
  - Must optimize something else!
- Empirical results: consistent improvement over all published P&R results
  - Routability, routed wirelength, via counts
HPWL vs. Steiner Tree WL

- HPWL $\leq$ Steiner Tree WL ( = for 2- and 3-pin nets)
- Computing HPWL takes linear time, but Steiner trees are NP-hard
- Steiner Tree tools we evaluate:
  - Batched Iterated 1-Steiner (BI1ST) [Kahng, Robins 1992]
    - Slow ($n^3$)
    - Very accurate, even for 20+ pins
  - FastSteiner [Kahng, Mandoiu, Zelikovsky 2003]
    - Faster but less accurate than BI1ST
  - FLUTE [Chu 2004, 2005]
    - Very fast
    - Optimal lookup tables for $\leq 9$ pins, less accurate for 10+ pins
Existing Placement Framework

- Consider *placement bins*
- Partition them
  - Use min-cut bisection
  - Place end-cases optimally
- *Propagate terminals* before partitioning
  - Terminals: fixed cells or cells outside current bin
  - Assigned to one of partitions
- Save runtime: a 20-pin may “propagate” into 3-pin net
  - “Inessential nets”: fixed terminals in both partitions (can be entirely ignored)
- Traditional min-cut placement tracks HPWL
Better Modeling of HPWL by Net Weights In Min-cut

- Introduced in Theto placer [Selvakkumaran 2004]
- Refined in [Chen 2005]
  - Shown to accurately track HPWL
- Use 1 or 2 hyper-edges to represent each net for partitioning
  - Weights given by costs $w_{\text{left}}$, $w_{\text{right}}$, $w_{\text{cut}}$
  - $w_{\text{left}}$: HPWL when all cells on left side (a)
  - $w_{\text{right}}$: HPWL when all cells on the right (b)
  - $w_{\text{cut}}$: HPWL when cells on both sides (c)

![Diagram showing net weights and their corresponding HPWL calculations.](image)

Fig. 2. An example of determining a net weight. (a), (b), and (c) are three possible partitioning results. (d), (e), and (f) are corresponding partitioning hypergraphs.

Figure from [Chen, Chang, Lin 2005]
Key Observation

For bisection, cost of each net is characterized by 3 cases

- Cost of net when cut: $w_{\text{cut}}$
- Cost of net when entirely in left partition: $w_{\text{left}}$
- Cost of net when entirely in right partition: $w_{\text{right}}$

In our work, we compute these costs for a different placement objective

- Real difficulty in data structures!
Our Contributions

- Optimization of Steiner WL
  - In global placement (runtime penalty ~30%)
  - In detail placement
- Whitespace allocation to tame congestion
- Empirical evaluation of ROOSTER
  - No violations on 16 IBMv2 benchmarks (easy + hard)
  - Consistent improvements of published results
  - 4-10% by routed wirelength
  - 10-15% by via counts
Optimizing Steiner WL During Global Placement

- Recall – each net can be modeled by 3 numbers
  - This has only been applied to HPWL optimization
- We calculate $w_{\text{left}}$, $w_{\text{right}}$, $w_{\text{cut}}$ using Steiner evaluator
  - For each net, before partitioning starts
  - The bottleneck is still in partitioning
    → can afford a fast Steiner-tree evaluator
- Pitfall : cannot propagate terminals!
  - Nets that were inessential are now essential
  - Must consider all pins of each net
  - More accurate modeling, but potentially much slower
New Data Structure for Global Placement

- **Pointsets with multiplicities**: two per net
- Unique locations of **fixed & movable** pins
  - At top placement layers, very few *unique* pin positions (except for fixed I/O pins)
- Maintain the number of pins at each location
  - Fast maintenance when pins get reassigned to partitions (or move)
- Allows to efficiently compute the 3 costs
Improvement in Global Placement

- Results depend on the Steiner tree evaluator
  - We choose FastSteiner (vs BI1ST and FLUTE)
  - See Appendix B for detailed comparison
- Impact of changes to global placement
  - Results consistent across IBMv2 benchmarks
  - Steiner WL reduction: 2.9%
  - HPWL grows by 1.3%
  - Runtime grows by 27%
Optimizing Steiner WL in Detail Placement

- We leverage the speed of FLUTE with two sliding-window optimizers
  - Exhaustive enumeration for 4-5 cells in a single row
  - Interleaving by dynamic programming (5-8 cells)
    - Fast but not always optimal
    - Using both reduces Steiner WL by 0.69%, routed WL by 0.72% and consumes 11.83% of [global + detail] placement runtime

- Much faster than single-trunk tree optimization from [Jariwala,Lillis 2004]
  - Our optimization seems stronger, not restricted to FPGAs
Congestion-based Cutline Shifting

- To reduce congestion ROOSTER allocates whitespace non-uniformly
- Based on the WSA technique [Li 2004]
  - WSA is applied after detail placement (our technique is used during global placement)
  - Identifies congested regions
  - Injects whitespace, causing cell overlap
  - Legalization and re-placement is required
  - Detail placement recovers HPWL
Congestion-based Cutline Shifting

- Our technique is applied pro-actively during mincut
  - No need for “re-placement” and legalization
  - This improves via counts

- Periodically, build up-to-date congestion maps
  - Use congestion maps from [Westra 2004]
  - Estimate congestion for each existing placement bin

- Cutlines shifted to equalize congestion in bins

<table>
<thead>
<tr>
<th>15% WS</th>
<th>15% WS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cong: 100</td>
<td>Cong: 200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10% WS</th>
<th>20% WS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cong: 150</td>
<td>Cong: 150</td>
</tr>
</tbody>
</table>
**Empirical Results: IBMv2**

ROOSTER: Rigorous Optimization Of Steiner Trees Eases Routing

**Published results:**

<table>
<thead>
<tr>
<th></th>
<th>Routed WL Ratio</th>
<th>Via Ratio</th>
<th>Routes with Violation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOSTER</td>
<td>1.000</td>
<td>1.000</td>
<td>0/16</td>
</tr>
<tr>
<td>mPL-R+WSA</td>
<td>1.055</td>
<td>1.156</td>
<td>0/16</td>
</tr>
<tr>
<td>APlace 1.0</td>
<td>1.042</td>
<td>1.119</td>
<td>1/8</td>
</tr>
<tr>
<td>Capo 9.2</td>
<td>1.056</td>
<td>Not published</td>
<td>0/16</td>
</tr>
<tr>
<td>Dragon 3.01</td>
<td>1.107</td>
<td>Not published</td>
<td>1/16</td>
</tr>
<tr>
<td>FengShui 2.6</td>
<td>1.093</td>
<td>Not published</td>
<td>7/16</td>
</tr>
</tbody>
</table>

**Most recent results:**

<table>
<thead>
<tr>
<th></th>
<th>Routed WL Ratio</th>
<th>Via Ratio</th>
<th>Routes with Violation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mPL-R+WSA</td>
<td>1.007</td>
<td>1.069</td>
<td>0/16</td>
</tr>
<tr>
<td>APlace 2.04</td>
<td>0.968</td>
<td>1.073</td>
<td>2/16</td>
</tr>
<tr>
<td>FengShui 5.1</td>
<td>1.097</td>
<td>1.230</td>
<td>10/16</td>
</tr>
</tbody>
</table>
ROOSTER with several detail placers: IBMv2

<table>
<thead>
<tr>
<th></th>
<th>Routed WL Ratio</th>
<th>Via Ratio</th>
<th>Routes with Violation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOSTER</td>
<td>1.000</td>
<td>1.000</td>
<td>0/16</td>
</tr>
<tr>
<td>ROOSTER+WSA</td>
<td>0.990</td>
<td>1.004</td>
<td>0/16</td>
</tr>
<tr>
<td>ROOSTER+Dragon 4.0 DP</td>
<td>1.041</td>
<td>1.089</td>
<td>2/16</td>
</tr>
<tr>
<td>ROOSTER+FengShui 5.1 DP</td>
<td>1.114</td>
<td>1.248</td>
<td>16/16</td>
</tr>
</tbody>
</table>
Improvement Breakdown: IBMv2 easy

Capo with uniform whitespace
optimizing StWL in global placement + above
congestion driven whitespace allocation + above
optimizing StWL in detailed placement (ROOSTER) + above

V = Violations
Improvement Breakdown: IBMv2 hard

Capo with uniform whitespace optimizing StWL in global placement + above
congestion driven whitespace allocation + above
optimizing StWL in detailed placement + above (ROOSTER)
Congestion with and without Capo -uniformWS

Capo -uniformWS
5 hours to route; 120 violations

ROOSTER
22 mins to route; 0 violations
Conclusions

- Steiner WL should be optimized in global and detail placement
  - Improves routability and routed WL
  - 10-15% improvement in via counts
  - Better Steiner evaluators may further reduce routed WL

- Congestion-driven cutline shifting in global placement is competitive with WSA
  - Better via counts
  - May be improved if better congestion maps available

- ROOSTER freely available for all uses
  [http://vlsicad.eecs.umich.edu/BK/PDtools](http://vlsicad.eecs.umich.edu/BK/PDtools)
Questions?
Huang & Kahng, ISPD1997

- Quadrisection can bias min-cut objective to Minimum Spanning Tree [Huang,Kahng 1997]
  - Loses accuracy by gridding terminals
  - 2x2 MST equivalent to 2x2 Steiner
  - Need much larger grid to truly optimize Steiner WL

- Compared to our work, Huang & Kahng…
  - Did not handle Steiner trees, only MSTs
    (handling Steiner trees may require 4x4 geometric partitioner)
  - Did not handle terminals very accurately
    (which seems to be the key)
  - Never evaluated the results with a router (!)