Critical–Trunk Based Obstacle–Avoiding Rectilinear Steiner Tree Routings for Delay and Slack Optimization

ISPD-2009

Yen-Hung Lin, Shu-Hsin Chang, Yih-Lang Li

Dept. of CSIE, National Chiao Tung University
OUTLINE

• Introduction
• Critical-Trunk Based Tree Growth
  • Performance-Driven OARST
  • Slack-Driven OARST
• Experimental Results
• Conclusions
INTRODUCTION

• OARST: Obstacle-Avoiding Rectilinear Steiner Tree
• Conventional OARST:
  • To minimize total wirelength
  • Maze-routing based manner
  • Spanning-graph based manner
    • Global view about pins and obstacles
    • Non-intersecting property
  • Minimization of wirelength may worsen the performance.
• Objectives:
  • To construct an OARST and consider delay simultaneously
  • To adopt routing algorithm in spanning-graph based manner
RELATION BETWEEN RADIUS AND DELAY

Figure 1. Relation between radius and delay in Shortest Path Tree (SPT)

Figure 2. Relation between radius and delay in Minimal Steiner Tree (MST)
Figure 3. Ideal tree topology for critical-trunk based tree growth
PERFORMANCE-DRIVEN OARST

- To minimize the maximum sink’s delay
- Overall flow of PDOARST:
  - Obstacle-Avoiding Spanning Graph Construction [11]
  - Performance-Driven Critical Trunk Growth
  - Performance-Driven Subtree Growth
  - Rectilinearization

PERFORMANCE-DRIVEN CRITICAL TRUNK GROWTH

- 2-pin net generation for routing algorithms
- Multi-source single-target maze routing

Figure 4. Performance-driven critical trunk growth

1. D
2. D
3. 5
4. D
5. 2

source window

D: Driver  S: Sink
: Obstacle corner
PERFORMANCE-DRIVEN CRITICAL TRUNK GROWTH

- Identification of performance-driven critical trunks
- PDCTF: Performance-driven criticality threshold factor
  \[ PDCTF = \frac{\text{average sink delay}}{\text{worst sink delay}} \]
- PDCR: Performance-driven Critical Radius
  \[ PDCR = PDCTF \times \max(\text{radius}) \]
- A sink is critical if its radius exceeds PDCR.
PERFORMANCE-DRIVEN SUBTREE GROWTH

- Delay penalty factor (PDF)
- To make the tree topology similar to the ideal one.

\[ DPF(i) = \begin{cases} 
\frac{R_i}{R_{\text{max}}}, & i \in N_{cr} \\
0, & \text{otherwise} 
\end{cases} \]

Figure 5. DPF and DPF inheritance
PERFORMANCE-DRIVEN SUBTREE GROWTH

• A* search like function
  • \( f(x) = g(x) + h(x) \)
  • \( g(x) = dist_{sx} + dist_{ds} \times DPF(s)^2 \times (1 - PDCTF) \)

Figure 6. The relation between performance-driven critical trunk & PDCTF

(a) PDCTF=0.854  (b) PDCTF=0.473
PERFORMANCE-DRIVEN vs. SLACK-DRIVEN

(a) Minimizing the maximum delay
(b) Satisfying the timing constraint

Figure 7. Steiner tree with different objective
SLACK-DRIVEN OARST

- To maximize the worst negative slack (WNS)
- Overall flow of SDOARST:

  - Obstacle-Avoiding Spanning Graph Construction [11]
  - Slack-Driven Critical Trunk Growth
  - Slack-Driven Subtree Growth
  - Rectilinearization
  - Redirection
SLACK-DRIVEN CRITICAL TRUNK GROWTH

- Sinks with smaller slacks prefer small delays.
- To guide 2-pin net generation with slack

Figure 8. Slack-driven critical trunk growth

\[ \text{dist}(p,q) \times \frac{\text{slack}(q) - \text{slack}_{\text{min}}}{\text{slack}_{\text{max}}} \]
SLACK-DRIVEN CRITICAL TRUNK GROWTH

- Identification of slack-driven critical trunks
- To compute priority of each sink
  - \( \text{priority}(i) = \text{slack}(i) - \text{delay}(i) \)
  - Small priority means that the attached sink has higher possibility to violate timing constraints.
- SDCP: Slack-Driven Critical Priority
  - Average priority of all sinks
SLACK-DRIVEN SUBTREE GROWTH

- Slack determines principally the allowable delay of a sink.
- Single-source single-target maze routing

Figure 9. Slack-driven subtree growth

D : Driver
: Sink
: Obstacle corner

Timing constraint violation!

Institute of Computer Science and Engineering, National Chiao Tung University
REDIRECTION

Figure 10. Flow of redirection mechanism
EXPERIMENTAL RESULTS

- **Platforms**
  1. A PC with 2.1 GHz AMD Athlon 64 Dual Core CPU and 1.5GB memory
  2. A workstation with 1.2 GHz CPU and 4GB memory

- **Benchmarks**

<table>
<thead>
<tr>
<th>Case</th>
<th>Pin</th>
<th>Obs</th>
<th>Case</th>
<th>Pin</th>
<th>Obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc01</td>
<td>10</td>
<td>10</td>
<td>rc07</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td>rc02</td>
<td>30</td>
<td>10</td>
<td>rc08</td>
<td>200</td>
<td>800</td>
</tr>
<tr>
<td>rc03</td>
<td>50</td>
<td>10</td>
<td>rc09</td>
<td>200</td>
<td>1000</td>
</tr>
<tr>
<td>rc04</td>
<td>70</td>
<td>10</td>
<td>rc10</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>rc05</td>
<td>100</td>
<td>10</td>
<td>rc11</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>rc06</td>
<td>100</td>
<td>500</td>
<td>rc12</td>
<td>1000</td>
<td>10000</td>
</tr>
</tbody>
</table>
ROUTING BASED TREE CONSTRUCTION

• To simplify the PDOARST only considering wirelength

• To compare with works which minimize the total wirelength


ROUTING BASED TREE CONSTRUCTION

Table 2. Comparison between wirelength and runtimes

| Case  | Wirelength | | | | Runtime |
|-------|------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| rc01  | 26810      | 0.34            | 0.34            | 2.72            | 0.01           | 0.00            | 0.00            | 0.00            |
| rc02  | 42280      | 1.32            | -0.17           | -0.59           | 0.01           | 0.00            | 0.00            | 0.00            |
| rc03  | 56160      | 0.50            | -0.73           | -0.16           | 0.01           | 0.00            | 0.00            | 0.00            |
| rc04  | 60710      | 0.21            | -0.59           | -1.91           | 0.01           | 0.00            | 0.00            | 0.00            |
| rc05  | 77330      | -0.47           | -1.29           | -1.31           | 0.01           | 1.00            | 0.00            | 0.00            |
| rc06  | 86299      | 0.12            | -3.40           | 1.31            | 0.06           | 3.50            | 0.83            | 0.33            |
| rc07  | 116801     | 0.54            | -3.03           | -0.90           | 0.06           | 5.50            | 1.17            | 0.17            |
| rc08  | 123004     | 0.29            | -3.46           | 1.50            | 0.09           | 5.22            | 2.33            | 0.67            |
| rc09  | 120062     | -0.26           | -3.24           | 0.41            | 0.15           | 3.80            | 1.53            | 0.60            |
| rc10  | 170600     | 0.50            | 0.05            | -1.02           | 0.03           | 6.67            | 5.33            | 0.00            |
| rc11  | 238905     | -0.33           | -0.96           | -1.30           | 0.10           | 5.40            | 7.60            | -0.20           |
| rc12  | 858310     | -1.72           | -8.06           | -0.69           | 2.89           | 14.39           | 19.23           | 0.37            |

| Ave.  | 3.40       | -2.05           | -0.01           | 3.79            | 3.17            | 0.16            |

- [*] diff of wirelength = ([*]-ours)/ours ×100.
- [*] spdup of runtime = [*]/ours-1.
## PERFORMANCE-DRIVEN OARST

### Table 3. Comparison between wirelength, worst delays, and runtimes of our simplified OARST and PDOARST

<table>
<thead>
<tr>
<th>Case</th>
<th>Wirelength</th>
<th>WorstDelay</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OARST (um)</td>
<td>PD diff (%)</td>
<td>OARST (ps)</td>
</tr>
<tr>
<td>rc01</td>
<td>26810</td>
<td>8.69</td>
<td>3709.40</td>
</tr>
<tr>
<td>rc02</td>
<td>42280</td>
<td>-0.50</td>
<td>4757.91</td>
</tr>
<tr>
<td>rc03</td>
<td>56160</td>
<td>8.53</td>
<td>8906.42</td>
</tr>
<tr>
<td>rc04</td>
<td>60710</td>
<td>11.55</td>
<td>8124.20</td>
</tr>
<tr>
<td>rc05</td>
<td>77330</td>
<td>12.17</td>
<td>11690.13</td>
</tr>
<tr>
<td>rc06</td>
<td>86299</td>
<td>5.19</td>
<td>10685.59</td>
</tr>
<tr>
<td>rc07</td>
<td>116801</td>
<td>5.19</td>
<td>13450.84</td>
</tr>
<tr>
<td>rc08</td>
<td>123004</td>
<td>4.71</td>
<td>16169.9</td>
</tr>
<tr>
<td>rc09</td>
<td>120062</td>
<td>12.62</td>
<td>20957.15</td>
</tr>
<tr>
<td>rc10</td>
<td>170600</td>
<td>2.87</td>
<td>25946.16</td>
</tr>
<tr>
<td>rc11</td>
<td>238905</td>
<td>2.37</td>
<td>36459.46</td>
</tr>
<tr>
<td>rc12</td>
<td>858310</td>
<td>39.84</td>
<td>464903.00</td>
</tr>
<tr>
<td>Ave.</td>
<td>9.44</td>
<td>-24.12</td>
<td>1.59</td>
</tr>
</tbody>
</table>
SLACK-DRIVEN OARST

Table 4. Comparison between wirelength, worst delays, WNSs, and runtimes of PDOARST and SDOARST

<table>
<thead>
<tr>
<th>Case</th>
<th>Wirelength</th>
<th>WorstDelay</th>
<th>WNS</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PD (um)</td>
<td>SD diff (%)</td>
<td>PD (ps)</td>
<td>SD diff (%)</td>
</tr>
<tr>
<td>rc01t</td>
<td>29140</td>
<td>3.88</td>
<td>3383.65</td>
<td>-9.08</td>
</tr>
<tr>
<td>rc02t</td>
<td>42070</td>
<td>14.67</td>
<td>4717.57</td>
<td>-3.87</td>
</tr>
<tr>
<td>rc03t</td>
<td>60590</td>
<td>7.66</td>
<td>5708.50</td>
<td>20.82</td>
</tr>
<tr>
<td>rc04t</td>
<td>67720</td>
<td>15.39</td>
<td>5540.01</td>
<td>5.18</td>
</tr>
<tr>
<td>rc05t</td>
<td>86740</td>
<td>12.23</td>
<td>7038.70</td>
<td>-2.42</td>
</tr>
<tr>
<td>rc06t</td>
<td>90777</td>
<td>34.33</td>
<td>10365.80</td>
<td>-8.64</td>
</tr>
<tr>
<td>rc07t</td>
<td>122858</td>
<td>44.67</td>
<td>11744.40</td>
<td>7.78</td>
</tr>
<tr>
<td>Rc08t</td>
<td>128803</td>
<td>41.07</td>
<td>12634.00</td>
<td>3.38</td>
</tr>
<tr>
<td>rc09t</td>
<td>135215</td>
<td>31.70</td>
<td>16769.19</td>
<td>-11.82</td>
</tr>
<tr>
<td>rc10t</td>
<td>175500</td>
<td>18.83</td>
<td>17319.80</td>
<td>-12.49</td>
</tr>
<tr>
<td>rc11t</td>
<td>244572</td>
<td>17.09</td>
<td>30291.00</td>
<td>-32.79</td>
</tr>
<tr>
<td>rc12t</td>
<td>1200290</td>
<td>18.69</td>
<td>164655.00</td>
<td>-4.54</td>
</tr>
</tbody>
</table>

Ave. | 21.68      | -4.04      | **83.19** | **-0.02** |

Institute of Computer Science and Engineering, National Chiao Tung University
CONCLUSIONS

- We apply an routing algorithm to construct Steiner tree in spanning-graph based approach.
- We propose a critical-trunk based tree growth mechanism.
- We construct an obstacle-avoiding rectilinear Steiner tree with different objective.
  - Minimization of maximum delay
  - Maximization of the worst negative slack