A Metal-Only-ECO Solver for Input-Slew and Output-Loading Violations

Chien-Pang Lu, Mango C.-T. Chao, Chen-Hsing Lo, and Chih-Wei Chang

Mstar Semiconductor, ChuPei, Taiwan
Dept. of EE, Nat’l Chiao Tung Univ., Hsinchu, Taiwan
Outline

- Metal-only ECO & its challenges
- Problem Formulation
- Proposed Slew>Loading-Violation Solver (MOESS)
  - Overall Flow
  - Increase spare-buffer pool
  - Wire-loading estimation
  - ESB mode (minimize # of inserted buffers)
  - ECT mode (reduce critical path’s delay)
- Experimental result
- Conclusions
Metal-Only ECO

- The increasing pressure of time-to-market has forced IC design houses to improve capability of handling incremental design changes
- Those design changes are often requested after silicon chips are manufactured
  - its photomasks need to be changed
- Solution: metal-only ECO
  - change only the metal layers (for interconnect) while the base layers (for cells) remain the same
  - reduce cost by reusing base-layer photomasks
  - shorten tape-out turn-around time
EDA Tools Needed in Metal-Only ECO

- Allocate spare cells all over a chip
  - EDA vendors already provide effective solutions
- Obtain netlist difference and implement the difference
  - EDA vendors already provide effective solutions
- A router dealing with a lot obstacles
  - EDA vendors already provide effective solutions
- Solve violations of timing-related factors, such as setup time, input slew, and output loading
  - However, vendor’s solutions are not effective so far
Outline

- Metal-only ECO & its challenges
- **Problem Formulation**
- Proposed Slew/Loading-Violation Solver (MOESS)
  - Overall Flow
  - Increase spare-buffer pool
  - Wire-loading estimation
  - ESB mode (minimize # of inserted buffers)
  - ECT mode (reduce critical path’s delay)
- Experimental result
- Conclusions
Problem Formulation of Proposed Work

- **Given:**
  - Input-slew and output-loading constraints
  - Nets violating the constraints after the design changes are implemented
  - Available spare cells

- **Objective**
  - Insert fewest spare cells as buffers to eliminate the violations

- Use a commercial APR tool to realize the buffer insertions

- Focus on how to select proper spare cells and estimate the added wire loading when inserting the buffers by the adopted APR tool
Transfer Input-slew Constraint into Equivalent Output-loading Constraint

- **OAL\(_g\)**: Output Available Loading
  - The maximum output loading of gate \(g\) which can generate an output slew smaller than the slew constraint assuming that \(g\)'s input slew is equal to slew constraint

- Obtaining **OAL\(_g\)** for each type of gate
  - Binary search, table look-up
    - Ex: target input slew constraint, 500ps

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Input slew</th>
<th>Output load</th>
<th>Output slew</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>500p</td>
<td>4000ff</td>
<td>2000p</td>
</tr>
<tr>
<td>2</td>
<td>500p</td>
<td>1500ff</td>
<td>400p</td>
</tr>
<tr>
<td>3</td>
<td>500p</td>
<td>1600ff</td>
<td>520p</td>
</tr>
<tr>
<td>4</td>
<td>500p</td>
<td><strong>1540ff</strong></td>
<td><strong>500p</strong></td>
</tr>
</tbody>
</table>

- **OAL(1540ff)**: under 500p input slew
Outline

- Metal-only ECO & its challenges
- Problem Formulation
- Proposed Slew/Loading-Violation Solver (MOESS)
  - Overall Flow
  - Increase spare-buffer pool
  - Wire-loading estimation
  - ESB mode (minimize # of inserted buffers)
  - ECT mode (reduce critical path’s delay)
- Experimental result
- Conclusions
Overall Flow of MOESS

1. Collect usable spare gates

2. For each slew/cap violation pin, apply ESB buffer-insertion scheme to solve the violation with least # of spare cells

3. For each timing violation net, apply ECT buffer-insertion scheme to reduce set-up time while satisfying slew/cap constraints

4. For each unsolved timing violation net, enforce priority routing using top metal or double spacing

Check STA timing report

OK

No

OK

No

Done
Increase Spare-Buffer Pool

- Recycle of redundant cells
  - APR tools use special tags to identify spare buffers
  - Tags may be lost by engineer’s incorrect operation
  - MOESS applies a breadth-first search starting from each floating output to recycle the lost-tag gates
- Function cells as buffers by connecting the other inputs to a constant
Wire-Loading Estimation for a Two-Terminal Net

- Use a net’s Manhattan distance ($MD$) to estimate its wire loading ($WL$)
- $WL(p1,p2) = MD_h(p1,p2) \times RRMD_h(VD(p1,p2) \times K_h + MD_v(p1,p2) \times RRMD_v(VD(p1,p2) \times K_v$}

Manhattan distance

Wire loading constant per routing unit

# of vias over rectangle area formed by p1 and p2

Routing ratio to Manhattan distance

This function is actually the average statistics collected from the past usage of the adopted ARP tool.
Solving Violation for a High-Fanout Net

- How to use fewest buffers to solve a violation?
  - How many terminals driven by a buffer?
Flow of ESB Buffer-Insertion Scheme
(Use fewest buffers to solve the violation)

A. Obtain MC (minimum-chain) order of net’s terminal pins

B. Group terminal pins based on the MC order

C. Calculate the ideal buffer location for driving grouped terminals

D. Search real spare buffer and insert it to the net

E. Update net and recalculate Its MC order

Meet loading constraint?

NO

OK

Done
Minimum-Chain Order of a Net’s Terminals

- **Algorithm:**
  - Start from violation gate
  - Select the closest terminal as the next ordered terminal until all terminal are ordered
Group Terminals

- Group the terminals based on the reversed MC order
- Each time add one terminal into the group
- Stop when adding the new terminal would exceed gate’s $OAL$ (slew/loading constraint)
  \[ \sum_{i=1}^{n} (InC_{p_{i}} + WL(p_{i},p_{i-1})) > OAL_g \]
- Use a buffer to drive as many terminals as possible
Calculate Ideal Location of Inserted Buffer

- Two rules when deciding ideal buffer’s location
  - **R1**: Use all buffer’s driving capability under the given constraint
    - \[ |X_b - X_p| * U_h(b, p_n) + |Y_b - Y_p| * U_v(b, p_n) \leq ORL_b \]
    - ORL - Output Remain Loading
      - \[ ORL_b = OAL_b - \sum_{i=1}^{n} (Inc_{p_i} + WL(p_i, p_{i-1})) \]
  - \( U_h \) and \( U_v \) are vertical and horizontal distance per loading unit
  - **R2**: Locate the inserted buffer as close to the violation gate as possible
    - \( \frac{Y_b - Y_p}{X_b - X_p} = \frac{Y_p - Y_g}{X_p - X_g} \)
    - Limit the ideal location between \( g \) and \( p_n \)
Find Spare Buffer near Ideal Location

- Use $ORL_b$ as the radius to draw the boundary of searching feasible spare buffers.
Backward Tolerance: Enlarge Searching Space

- Ungroup the last grouped terminal to increase the ORL and in turn the radius of the search space
  - Keep on ungrouping until a spare gate is found

\[
\begin{align*}
\text{\(\triangleright\)}: & \text{ FG, farthest group} \\
\text{\(\triangleright\triangleright\)}: & \text{ spare not candidate} \\
\text{\(\triangleright\triangleright\triangleright\)}: & \text{ candidate spare gate}
\end{align*}
\]
Example

Output loading exceed constraints
Update net and recalculate its MC order

Output loading meet constraints.
Done!
ECT Mode (reduce critical path’s delay)

- Separate the grouping of original terminals from the grouping of new-added terminals far away from the violation output
  - Group the distant, new-added terminals first

ESB Mode

ECT Mode

extra delay from buffer!

 timing critical terminal

MD constraint $\Rightarrow 1000\mu s$

After ECO multi-cycle path

After ECO multi-cycle path
Outline

- Metal-only ECO & its challenges
- Problem Formulation
- Proposed Slew/Loading-Violation Solver (MOESS)
  - Overall Flow
  - Increase spare-buffer pool
  - Wire-loading estimation
  - ESB mode (minimize # of inserted buffers)
  - ECT mode (reduce critical path’s delay)
- Experimental result
- Conclusions
# Design Information

<table>
<thead>
<tr>
<th>Proj. (ver.)</th>
<th>inst. count</th>
<th>process</th>
<th>spare count</th>
<th>ECO size</th>
<th>violation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>slew</td>
</tr>
<tr>
<td>Da(3)</td>
<td>190K</td>
<td>.18</td>
<td>7.6K</td>
<td>142</td>
<td>40</td>
</tr>
<tr>
<td>Db(3)</td>
<td>210K</td>
<td>.18</td>
<td>9.1K</td>
<td>1030</td>
<td>6</td>
</tr>
<tr>
<td>Dc(4)</td>
<td>242K</td>
<td>.18</td>
<td>5.5K</td>
<td>507</td>
<td>71</td>
</tr>
<tr>
<td>Dd(3)</td>
<td>309K</td>
<td>.18</td>
<td>10.4K</td>
<td>1904</td>
<td>47</td>
</tr>
<tr>
<td>De(2)</td>
<td>871K</td>
<td>.13</td>
<td>62.4K</td>
<td>127</td>
<td>0</td>
</tr>
<tr>
<td>Df(2)</td>
<td>1.3M</td>
<td>.13</td>
<td>48.8K</td>
<td>1276</td>
<td>15</td>
</tr>
<tr>
<td>Dg(4)</td>
<td>1.6M</td>
<td>.13</td>
<td>80.5K</td>
<td>1702</td>
<td>166</td>
</tr>
</tbody>
</table>
## Experiment Result

<table>
<thead>
<tr>
<th>Proj. (ver.)</th>
<th>worst slew</th>
<th>worst loading</th>
<th>worst slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Da(3)</td>
<td>5.0n</td>
<td>1.9n</td>
<td>&lt;1</td>
</tr>
<tr>
<td>Db(3)</td>
<td>1.8n</td>
<td>1.8n</td>
<td>&lt;1</td>
</tr>
<tr>
<td>Dc(4)</td>
<td>3.8n</td>
<td>2.0n</td>
<td>&lt;1</td>
</tr>
<tr>
<td>Dd(3)</td>
<td>2.1n</td>
<td>2.0n</td>
<td>&lt;1</td>
</tr>
<tr>
<td>De(2)</td>
<td>0.9n</td>
<td>0.9n</td>
<td>1.2</td>
</tr>
<tr>
<td>Df(2)</td>
<td>1.3n</td>
<td>0.9n</td>
<td>3.5</td>
</tr>
<tr>
<td>Dg(4)</td>
<td>1.2n</td>
<td>1.0n</td>
<td>4.6</td>
</tr>
</tbody>
</table>

* means the result violates the constraint
Experiment Result

# of inserted spare buffer (average imp. 38%)

Run Time (average 14.9X faster)
## Experiment Result

<table>
<thead>
<tr>
<th>Proj. (ver.)</th>
<th>spare count</th>
<th>ECO size</th>
<th>#violation</th>
<th>worst slew</th>
<th>worst loading</th>
<th>worst slack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>slew</td>
<td>[3] MOESS</td>
<td>MOESS</td>
</tr>
<tr>
<td>Dh(2)</td>
<td>4.2K</td>
<td>52</td>
<td>7</td>
<td>0</td>
<td>1.8n</td>
<td>&lt;1</td>
</tr>
<tr>
<td>Dh(3)</td>
<td>4.1K</td>
<td>267</td>
<td>29</td>
<td>2</td>
<td>2.1n</td>
<td>&lt;1</td>
</tr>
<tr>
<td>Dh(5)</td>
<td>3.7K</td>
<td>1672</td>
<td>118</td>
<td>5</td>
<td>4.6n</td>
<td>1.3</td>
</tr>
<tr>
<td>Dh(7)</td>
<td>1.9K</td>
<td>43</td>
<td>9</td>
<td>2</td>
<td>2.8n</td>
<td>&lt;1</td>
</tr>
<tr>
<td>Dh(8)</td>
<td>1.8K</td>
<td>135</td>
<td>17</td>
<td>3</td>
<td>3.5n</td>
<td>&lt;1</td>
</tr>
</tbody>
</table>

# of instance count: 352.1K

[Green box] means the result violates the constraint
Experiment Result

**# of inserted spare buffer**

- dh(2): 127 [3], 12 [MOESS]
- dh(3): 64 [3], 35 [MOESS]
- dh(5): 188 [3], 124 [MOESS]
- dh(7): 1813 [3], 29 [MOESS]
- dh(8): 29 [3], 19 [MOESS]

**Run Time**

(average speed up 29.9X)

- dh(2): 35 [3], 35 [MOESS]
- dh(3): 39 [3], 39 [MOESS]
- dh(5): 42 [3], 4.5 [MOESS]
- dh(7): 35 [3], 1 [MOESS]
- dh(8): 36 [3], 1 [MOESS]
Conclusions

• An effective solver to solve the slew/loading violations generated in metal-only ECO
• Less # of spare gates in use
• Shorter runtime
• The proposed solver can be ported to other APR tools as long as the tools can provide open access to its design database
Thank you!
Solving Slew>Loading Violation in Metal-Only ECO

- Current commercial tools are not aware of the physical locations of spare gates
  - Available spare gates may deviate from its ideal location

Lack of physical knowledge!