TRECO: Dynamic Technology Remapping for Timing Engineering Change Orders

Kuan-Hsien Ho, Jie-Hong R. Jiang, and Yao-Wen Chang

Graduate Institute of Electronics Engineering
National Taiwan University
Taipei, Taiwan
Outline

- Engineering Change Orders
- Problem Formulation
- Design Flow & Algorithms
  - Table Generation
  - Cut Enumeration
  - Table Lookup
  - Technology Remapping
- Timing-Aware Functional ECO
- Experimental Results
- Conclusions

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Gate count</th>
<th># Spare Cells</th>
<th>Check Period (ns)</th>
<th># ECO Paths</th>
<th># Critical Gates</th>
<th>WNS (ns)</th>
<th>TNS (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design_1</td>
<td>10542</td>
<td>569</td>
<td>41</td>
<td>16</td>
<td>9958</td>
<td>4.43</td>
<td>31.99</td>
</tr>
<tr>
<td>Design_2</td>
<td>31154</td>
<td>1049</td>
<td>68</td>
<td>10</td>
<td>1073</td>
<td>7.24</td>
<td>35.83</td>
</tr>
<tr>
<td>Design_3</td>
<td>1057</td>
<td>624</td>
<td>22</td>
<td>96</td>
<td>3675</td>
<td>1.87</td>
<td>98.82</td>
</tr>
<tr>
<td>Design_4</td>
<td>77297</td>
<td>2919</td>
<td>30</td>
<td>32</td>
<td>11979</td>
<td>2.75</td>
<td>58.83</td>
</tr>
<tr>
<td>Design_5</td>
<td>30825</td>
<td>839</td>
<td>14</td>
<td>38</td>
<td>1984</td>
<td>0.55</td>
<td>8.37</td>
</tr>
</tbody>
</table>
Engineering Change Orders (ECOs)

• ECOs are for incremental changes of a design

• In view of optimization stages:
  – Pre-mask ECOs
    ■ Done before chip-mask being manufactured
  – Post-mask ECOs
    ■ Done after chip-mask being manufactured

• In view of optimization goals
  – Functional ECOs
    ■ Applied for functional rectification
  – Timing ECOs
    ■ Applied for timing repair
Engineering Change Orders (ECOs)

Functional ECOs

Pre-mask ECOs

Timing ECOs

Post-mask ECOs
Engineering Change Orders (ECOs)

- ECOs are for incremental changes of a design
- Generally, ECOs can be divided into two categories,

  - Pre-mask ECOs
    - Done before chip-mask being manufactured
    - Usually for design-time saving

  - Post-mask ECOs
    - Done after chip-mask being manufactured
    - Usually for design-cost saving

- Spare cells are redundant cells reserved in a chip design and mainly used for post-mask ECOs
More on Post-Mask ECOs

• Since masks for transistors are much more expensive than those for metal layers,
  – Metal-only ECO, which performs design changes on metal layers, is the mainstream

• To enable metal-only ECO, spare cells are spread over the circuit layout after placement

Previous Work

• Functional ECOs
  – Technology remapping considering wirelength minimization

• Timing ECOs
  – Gate sizing, buffer insertion, and/or technology remapping
Timing-Aware Functional ECO Problem

• Given
  – A placed and routed circuit netlist
  – A set of placed spare cells
  – A cell library
  – A timing constraint
  – A set of correction logic netlists for functional rectification

• Our goal is to rewire the design such that
  – The whole circuit functionality is corrected
  – The timing constraint is satisfied

• When functional rectification is not needed, the problem reduces to timing ECO problem
Design Flow

Pre-processing
- Table Generation
- Placed Netlist, Cell Library, and Timing Constraints
- Static Timing Analysis

Optimization
- Critical Gate Selection
- Table Lookup
- Technology Remapping
- Incremental Update

- Convergence
- Output Result
Design Flow

Pre-processing

Table Generation

Placed Netlist, Cell Library, and Timing Constraints

Static Timing Analysis

Critical Gate Selection

Table Lookup

Technology Remapping

Incremental Update

Convergence

Output Result

Violation Y

Cut Enumeration

N
Two functions are in the same **NPN equivalence class** if they can be equivalent by

- **Negation of any variable(s)**
- **Permutation of variables and/or**
- **Negation of the function**

ex. \( F = a + bc \)
\( G = b + ac \)  \( \begin{cases} \quad F \text{ and } G \text{ are NPN-equivalent} \end{cases} \)

<table>
<thead>
<tr>
<th># inputs</th>
<th># Functions</th>
<th># NPN classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>256</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>65,536</td>
<td>222</td>
</tr>
<tr>
<td>5</td>
<td>4,294,967,296</td>
<td>616,126</td>
</tr>
</tbody>
</table>

222 pattern graphs are enough to form all 4-input Boolean functions

Our pattern graphs are composed of NAND, NOR, and inverters
To enhance the diversity of pattern graphs, we further construct a table called PLG-subtable.

All gate types in the given cell library are considered.
Look up Table Overview

F = a+bc
F = abc

NPN classes (222 slots)

Primitive logic gates

Look up Table

NPN subtable

PLG subtable
Design Flow

Pre-processing

Table Generation

Placed Netlist, Cell Library, and Timing Constraints

Static Timing Analysis

Optimization

Critical Gate Selection

Table Lookup

Technology Remapping

Incremental Update

Convergence

Cut Enumeration

Violation

Output Result
A set of gates $C$ is a cut for a gate $g$ if
- Each path from primary inputs (PIs) to $g$ must go through at least one node of $C$.

A cut is called $k$-feasible if
- # of gates in $C$ does not exceed the bound, $k$. 

**Cut Enumeration**

![Diagram showing cut points in a circuit]
Design Flow

Pre-processing
- Table Generation
- Placed Netlist, Cell Library, and Timing Constraints
- Static Timing Analysis

Optimization
- Critical Gate Selection
- Table Lookup
- Technology Remapping
- Incremental Update

Convergence
- Y
- N

Cut Enumeration
- Y
- N

Violation
- Y
- N

Output Result
Table Lookup

A critical gate

Spare cell collection

Few # of NAND & NOR gates?

Yes

PLG-subtable lookup

No

Cut extraction

Functionality analysis

NPN-subtable lookup
Design Flow

Pre-processing
- Table Generation
- Placed Netlist, Cell Library, and Timing Constraints
  - Static Timing Analysis

Optimization
- Critical Gate Selection
  - Table Lookup
  - Technology Remapping
  - Incremental Update
  - Convergence

Y Y N N

Violation
- Cut Enumeration

Output Result
Spare-cell selection problem

In the following example

- \( C_2^4 \) choices
- Complexity: \( O(n^2) \) if there are \( n \) spare cells
Challenges of Remapping (2/2)

- Even if the same spare cells are chosen...

Larger wiring cost

Smaller wiring cost

The wiring cost might be quite different!
Algorithm of Technology Remapping

Rip-up the target sub-circuit
Map to a new circuit with a given pattern graph randomly
Reduce total wirelength
Optimize timing of the new circuit
Remapping – Rip-Up & Map

pattern graph

cut

Iterative wirelength reduction
Iterative timing optimization

Rip-up & Map
Remapping – Wirelength Reduction

- Rip-up & Map
- Iterative wirelength reduction
- Iterative timing optimization

Pattern graph
Remapping – Timing Optimization

- Rip-up & Map
- Iterative wirelength reduction
- Iterative timing optimization

pattern graph

cut
Extension: Timing-Aware Functional ECO

• By NPN-subtable and the remapping techniques,

from NPN-subtable
Experimental Setup

- C++ programming language
- Linux workstation
  - 2.4 GHz CPU and 8 GB memory
- Five industrial designs using 180nm technology

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Gate count</th>
<th># Spare Cells</th>
<th>Clock Period (ns)</th>
<th># ECO Paths</th>
<th># Critical Gates</th>
<th>WNS (ns)</th>
<th>TNS (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design_1</td>
<td>19342</td>
<td>569</td>
<td>41</td>
<td>16</td>
<td>9938</td>
<td>4.43</td>
<td>31.99</td>
</tr>
<tr>
<td>Design_2</td>
<td>31154</td>
<td>1049</td>
<td>68</td>
<td>10</td>
<td>1073</td>
<td>7.24</td>
<td>35.83</td>
</tr>
<tr>
<td>Design_3</td>
<td>10657</td>
<td>624</td>
<td>22</td>
<td>96</td>
<td>3675</td>
<td>1.87</td>
<td>98.82</td>
</tr>
<tr>
<td>Design_4</td>
<td>77297</td>
<td>2919</td>
<td>30</td>
<td>32</td>
<td>11979</td>
<td>2.75</td>
<td>58.83</td>
</tr>
<tr>
<td>Design_5</td>
<td>30825</td>
<td>839</td>
<td>14</td>
<td>38</td>
<td>1984</td>
<td>0.55</td>
<td>8.37</td>
</tr>
</tbody>
</table>

- The spare-cell types considered were AND, OR, NAND, NOR, buffer, and inverter
Experimental Results

- The execution time of TRECO depends on several factors, such as
  - # of critical gates,
  - Cut size and cut volume of each critical gate,
  - # of subgraphs being considered,
  - Success rate and effectiveness of remapping on a gate,
  - Spare-cell density, etc.

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Initial</th>
<th>Technology Remapping Part of [4]</th>
<th>TRECO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># ECO Paths</td>
<td>WNS (ns)</td>
<td>TNS (ns)</td>
</tr>
<tr>
<td>Design_1</td>
<td>16</td>
<td>4.43</td>
<td>31.99</td>
</tr>
<tr>
<td>Design_2</td>
<td>10</td>
<td>7.24</td>
<td>35.83</td>
</tr>
<tr>
<td>Design_3</td>
<td>96</td>
<td>1.87</td>
<td>98.82</td>
</tr>
<tr>
<td>Design_4</td>
<td>32</td>
<td>2.75</td>
<td>58.83</td>
</tr>
<tr>
<td>Design_5</td>
<td>38</td>
<td>0.55</td>
<td>8.37</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td>0.73x</td>
<td>0.77x</td>
</tr>
</tbody>
</table>

Conclusions

• We have presented a post-mask ECO technique, called TRECO, which supports timing as well as functional ECOs.

• Compared with prior work, it offers a more effective and efficient solution on timing optimization using technology remapping.

• Experimental results based on five industrial designs show that TRECO can alleviate timing violations under reasonable CPU times.
Thank You!

Kuan-Hsien Ho
khho@eda.ee.ntu.edu.tw