Safe Delay Optimization for Physical Synthesis

Kai-hui Chang, Igor L. Markov and Valeria Bertacco
University of Michigan at Ann Arbor

Jan. 25, 2007

Critical path in a placement
Design: DES
Cell count: 89341

Improving Deep Submicron Layouts

- Current technology trends (< 90nm)
  - Interconnect delay is dominant
  - Impact of logic restructuring is difficult to predict
  - Use of placement information is critical

- Existing post-placement timing optimizations break down
  - Logic level timing analysis is inaccurate (route length and net delay estimates are arbitrary)
  - Estimated improvements may worsen timing
  - May increase congestion and route length
  - Critical nets may detour during routing
  - Lack of predictability

Our Work: Improving Predictability

- We define a new parameter of physical synthesis optimizations: physical safeness
- We propose a new safe physical synthesis technique
  - Predictable delay improvement
  - Easily verifiable correctness
  - Up to 86% improvement for IWLS2005 benchmarks with < 1% increase in route length and via count
  - 11% delay improvement on average

Outline

- Physical safeness
- Our physical synthesis approach
- Experimental results
- Conclusions
**Physical Safeness**

- Preserving physical parameters
  - Timing, congestions, distances, locations
- Safe techniques allow only legal changes
  - Accurate analysis can be performed
  - Changes that worsen layout are rejected immediately
- Unsafe techniques allow overlaps and route length increase
  - Legalization is required
  - Accurate analysis cannot be performed immediately

**Safe/Unsafe Examples: Rewiring**

- Symmetry-based rewiring
- Physically safe
- ATPG-based rewiring
- Physically unsafe

Must call legalizer to remove overlaps

**Physical Synthesis Techniques**

<table>
<thead>
<tr>
<th>Technique</th>
<th>Physical safeness</th>
<th>Optimization strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetry-based rewiring</td>
<td>Safe</td>
<td>Low</td>
</tr>
<tr>
<td>ATPG-based rewiring, buffer insertion, gate sizing, gate relocation</td>
<td>Unsafe</td>
<td>Low</td>
</tr>
<tr>
<td>Gate replication</td>
<td>Unsafe</td>
<td>Medium</td>
</tr>
<tr>
<td>Restructuring</td>
<td>Unsafe</td>
<td>High</td>
</tr>
<tr>
<td><strong>Safe Resynthesis</strong></td>
<td><strong>Safe</strong></td>
<td><strong>Medium</strong></td>
</tr>
</tbody>
</table>

Safe Resynthesis is useful by itself or after unsafe techniques for further optimization
**Outline**

- Physical safeness
- Our physical synthesis approach
  - Naïve variant
  - Enhanced variant
- Experimental results
- Conclusions

---

**Safe Resynthesis**

1. Simulate patterns and generate a signature for each wire in the circuit

<table>
<thead>
<tr>
<th>Input vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁, I₂, I₃, I₄, I₅</td>
</tr>
<tr>
<td>0 1 1 0 0</td>
</tr>
<tr>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>1 1 0 0 1</td>
</tr>
<tr>
<td>1 0 1 0 0</td>
</tr>
</tbody>
</table>

---

**Safe Resynthesis (Naïve Approach)**

2. Resynthesize the target wire \( w_I \) with combinations of different gates and wires

   - \( w_1 \) with \( g_1 \)
   - \( w_2 \) with \( g_n \)
   - \( w_3 \) with \( g_m \)
   - ... (naïve: may end up trying gates of all types at all locations)

---

**Safe Resynthesis**

3. Place the new gate at overlap-free sites near the center-of-gravity of its inputs and outputs

   - The location with the maximum improvement will be chosen
   - Equivalence checking verifies its correctness
**Safe Resynthesis (Faster Approach)**

- Trying all possible combinations is too expensive – will be improved
- Efficient search pruning
  1. Physical constraints: improves timing? (based on arrival times, locations and distances)
  2. Logical constraints: preserves functionality? (based on controlling values of gates)
- Resynthesis performed only if all constraints are satisfied

**Pruning 1: Physical Constraints**

1. Use arrival times (AT) from incremental STA
   - Consider only gates with $AT_{\text{gate}} < AT_{\text{target}}$
2. Try only gates close to the original driver

**Pruning 2: Logical Compatibility**

- Based on the controlling values of gates (AND, OR, but not XOR)
- Checks compatibility of all bits in
  - Candidate signature
  - Target signature
- Ignore wires with incompatible signatures
  - Reduce the number of candidate wires

**Implementation Insights**

- Accelerate compatibility test
  - “One-count”: number of 1s in the signature
  - E.g., one-count decreases for an AND gate
- Improve signature quality
  - Poor signatures require more equivalence checking
  - Uses patterns produced by the FRAIG package in ABC synthesis package (UCB)
    - Distinguish different wires in an AIG
Analysis of Our Approach

- Scalability
  - Signature + equivalence checking scales better than BDDs in terms of memory usage
  - Can handle 100K gate designs
- Optimization power
  - Utilizes complete controllability don’t-cares
  - Subsumes gate relocation and replication
  - Finds long range opportunities
- Safeness
  - Accurate analysis can be performed for each change

Outline

- Physical safeness
- Our physical synthesis approach
- Experimental results
- Conclusions

Experimental Setup

- Placer: Capo and QPlace
- Timing analyzer:
  - Before routing: D2M + Steiner tree
  - After routing: routed nets
- Benchmarks: IWLS2005, 0.18 µm library

<table>
<thead>
<tr>
<th>Suite</th>
<th>Benchmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenCores</td>
<td>SPI, DES_AREA, TV80, SYSTEMCAES, MEM_CTRL, AC97, USB, PCI, AES WB_CONMAX, Ethernet, DES_PERF</td>
</tr>
<tr>
<td>Faraday</td>
<td>DMA</td>
</tr>
<tr>
<td>ITC99</td>
<td>B14, B15, B17, B18, B22</td>
</tr>
<tr>
<td>ISCAS89</td>
<td>S35932, S38417</td>
</tr>
</tbody>
</table>

Our Experiments
Delay Improvement (30% Whitespace)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Estimated delay improvement</th>
<th>Routed delay improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC97</td>
<td>2.67%</td>
<td>3.67%</td>
</tr>
<tr>
<td>USB</td>
<td>5.21%</td>
<td>5.29%</td>
</tr>
<tr>
<td>PCI</td>
<td>5.99%</td>
<td>5.37%</td>
</tr>
<tr>
<td>AES</td>
<td>2.32%</td>
<td>5.06%</td>
</tr>
<tr>
<td>WB</td>
<td>61.37%</td>
<td>61.54%</td>
</tr>
<tr>
<td>Ether.</td>
<td>85.66%</td>
<td>86.41%</td>
</tr>
<tr>
<td>DES</td>
<td>1.98%</td>
<td>2.21%</td>
</tr>
<tr>
<td>Ave.</td>
<td>23.60%</td>
<td>24.22%</td>
</tr>
</tbody>
</table>

Improvement: unsafe+safe > safe > unsafe; unsafe resynthesis may worsen routed timing

Delay Improvement (Different Percentage of Whitespace)

<table>
<thead>
<tr>
<th>Percentage of whitespace</th>
<th>Estimated delay improvement</th>
<th>Routed delay improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>30%</td>
<td>23.60%</td>
<td>24.22%</td>
</tr>
<tr>
<td>10%</td>
<td>23.59%</td>
<td>24.12%</td>
</tr>
<tr>
<td>3%</td>
<td>20.33%</td>
<td>20.78%</td>
</tr>
</tbody>
</table>

- Safe and unsafe resynthesis have similar performance
- Unsafe+safe resynthesis achieves the most improvement

Route Length and Via Count Increase (Different Percentage of Whitespace)

<table>
<thead>
<tr>
<th>Percentage of whitespace</th>
<th>Route length increase</th>
<th>Via count increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>30%</td>
<td>0.08%</td>
<td>0.05%</td>
</tr>
<tr>
<td>10%</td>
<td>0.05%</td>
<td>0.09%</td>
</tr>
<tr>
<td>3%</td>
<td>0.04%</td>
<td>0.05%</td>
</tr>
</tbody>
</table>

Route length increase is small for all layouts, while via count increase is significant for layouts produced by unsafe resynthesis

Summary of Empirical Results

- 22% smaller delay at 30% whitespace
- 20% smaller delay at 3% whitespace
- Route length and via count increase < 1%
- Unsafe optimization
  - Provides better improvement before legalization and routing
  - Improvement after routing is hard to predict
  - Increases via count
- Safe optimization
  - Effects are more predictable
  - Does not increase via count
Conclusions

- Physical safeness
  - Effects of unsafe techniques are hard to evaluate
  - Safe techniques may provide better improvement
- A safe resynthesis technique
  - Up to 86% delay improvement
  - Route length and via count increase by less than 1%
- Unsafe + safe optimization leads to the most delay improvement
  - More powerful safe optimizations
  - Techniques to apply unsafe optimizations in a safe way