FPGA Interconnect Delay Fault Testing

Erik Chmelar
Center for Reliable Computing
Stanford University
April 7, 2003

Outline

- Introduction
- Previous Work
- Implementation
- Analysis
- Conclusion
- Future Work
Introduction

Motivation

• ~80% of die area is routing resources
  – Wire segments, vias, PIPs, PSMs
• Susceptible to resistive open defects
  – Resistive via or PIP, partially-open metal wire
• Resistive opens can lower reliability
  – Latent defect can ‘worsen’
    • Metal migration

Objective

• Devise testing method to detect delay faults
  – Also detect stuck-at 0/1, stuck-open, bridging
  – Few configurations or use partial reconfig.
    • Partial reconfiguration = fast test time
  – Ability to localize fault
    • Diagnosis
    • Application-specific FPGA model [Xilinx 03]
Introduction

Definitions

- Delay faults: rising \( \uparrow \) falling \( \downarrow \)

- Slice: smallest functional logic block
- Path: group of elements adjoining two slices
  - Wire segments, vias, PSMs, PIPs
- Set: group of paths simultaneously tested

Outline

- Introduction
- Previous Work
- Implementation
- Analysis
- Conclusion
- Future Work
Previous Work
[Tahoori 02]

- Configure chains of paths and logic blocks
  - Add additional load to path
    - Activate one or more additional PIPs
- Shift 1 (0) synchronously through chain
  - Determine where 1 (0) is ‘lost’
  - Delay $\alpha [R_{\text{tr}}(V_{\text{DD}}) + R_{\text{def}}][C_{\text{segment}} + C_{\text{load}}]$
    $= [R_{\text{segment}} + R_{\text{def}}][C_{\text{segment}} + C_{\text{load}}]$
  - Additional load increases delay of path
Previous Work
[Abr. & Str. 02]

• Configure chains of paths and logic blocks
  – Identical sequence of LUTs, latches, wire segments, etc. (all transparent, asynch)
• Apply input transition
• Measure phase difference at output by an oscillator loop and counter
  – Determine threshold number of oscillations
    • Less oscillations = pass
    • More oscillations = fail

Outline

• Introduction
• Previous Work
• Implementation
• Analysis
• Conclusion
• Future Work
Implementation

Overview

- Configure FPGA into set/slice chains
  - Logic configured in slices
  - Sets partitioned into X and Y paths
    - X and Y paths driven to opposite polarity
Implementation

Overview

- 2 slice configs. needed to test both $\overline{F}$ and $\overline{L}$
  - Config. A (Test Phase A): $X, Y$ $\overline{F}$
  - Config. B (Test Phase B): $X, Y$ $\overline{L}$
- Apply single input transition to each chain
  - Test Phase A: $\overline{F}$ Test Phase B: $\overline{L}$
- Observe single output transition
  - Test Phase A: $\overline{L}$ Test Phase B: $\overline{F}$

- Methodology bounded by current architectures
  - Constrain logic to smallest functional block
    - Xilinx = slice, Altera = LE (logic element)
    - ‘Extra’ routing is needed if logic requires multiple slices to connect logic together
Implementation
Overview
• Slice functionality (configured logic)
  – Detects if delay defect present in previous set of paths (slice inputs)
    • Logic creates race condition between paths
    • Signals arrive close in time = pass
    • Up to N-1 of N signals arrive late = fail
  – Controls next set of paths (slice outputs)
    • Pass = prop. Pass Signal Transition (PST)$^1$
    • Fail = prop. Fail Signal Transition (FST)$^2$

$^{1,2}$Defined in example on slides 21, 22

Implementation
Simplified View of a Slice or LE
• 2 logic elements (LUT/RAM/SR)
• 2 bistable elements (latch/flip flop)
Implementation

Starter Block (Test Phase A): $X \rightarrow \leftarrow$, $Y \rightarrow \leftarrow$

- 1 tester input signal to avoid skew problems
  - Test Phase B starter block similar

Configuration A (Test Phase A): $X \rightarrow \leftarrow$, $Y \rightarrow \leftarrow$

- $Clock_i$ is a delayed version of $First_i$
- Race condition between $Clock_i$ and $Last_i$
Implementation

Configuration B (Test Phase B): X \( \overline{\text{X}} \), Y \( \overline{\text{Y}} \)

- Partial reconfiguration of Configuration A
  - \( \text{Clock}_i \rightarrow \text{Clock}'_i \), + → &, init : 0 → 1, 1 → 0

Outline

- Introduction
- Previous Work
- Implementation
- **Analysis**
- Conclusion
- Future Work
Analysis
Below-threshold Delay Fault (Cfg. A)

Analysis
Above-threshold Delay Fault (Cfg. A)
Analysis

- Timing parameters
  - Average minimum feedback delay
    - $t_{feedback_{avg-min}}$
  - Max path delay difference guaranteed to pass
    - $t_{defect_{max-pass}} = t_{feedback} - t_{setup}$
  - Min path delay difference guaranteed to fail
    - $t_{defect_{min-fail}} = t_{feedback} + t_{hold}$
Analysis

Architecture-independent

- Min feedback delay is architecture-dependent
  - 360 ps (Spartan-IIE) – 1020 ps (Virtex)
  - But dependence doesn’t matter
    - Usually too small anyway (sometimes < 0)
      - Too sensitive => false failures
    - Allow margin for path delay mismatches
    - Use longer feedback path => less sensitive
- Slice/LE always has 2 LUTs, 2 FF/latches

Analysis

Resistive Open Defects

\[
V(t) = V_{DD} \left(1 - e^{-t/(R_{\text{segment}} + R_{\text{defect}})C_{\text{segment}}} \right)
\]

\[
t_{50\%} = -\left(R_{\text{segment}} + R_{\text{defect}}\right)C_{\text{segment}} \ln(0.5)
\]

\[
t_{\text{defect}} = t_{50\%_{\text{faulty}}} - t_{50\%_{\text{fault-free}}} = -R_{\text{defect}}C_{\text{segment}} \ln(0.5)
\]

\[
R_{\text{defect_{max-pass}}} = \frac{-t_{\text{defect_{max-pass}}}}{C_{\text{segment}} \ln(0.5)}
\]

\[
R_{\text{defect_{min-fail}}} = \frac{-t_{\text{defect_{min-fail}}}}{C_{\text{segment}} \ln(0.5)}
\]
Analysis

Maximum Sensitivity Bounds, Guaranteed Failure

<table>
<thead>
<tr>
<th>Device</th>
<th>Speed</th>
<th>Clock$_{eff}$ (GHz)</th>
<th>$R_{\text{defect-min-fail}}$ (kΩ) $(C_{\text{segment}} = .5 \text{ pF})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-II</td>
<td>fast</td>
<td>1.19</td>
<td>2.4</td>
</tr>
<tr>
<td></td>
<td>slow</td>
<td>1.19</td>
<td>2.4</td>
</tr>
<tr>
<td>Spartan-IIE</td>
<td>fast</td>
<td>2.78</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>slow</td>
<td>2.78</td>
<td>1.0</td>
</tr>
<tr>
<td>Virtex</td>
<td>fast</td>
<td>0.98</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>slow</td>
<td>0.98</td>
<td>2.9</td>
</tr>
<tr>
<td>Virtex-II</td>
<td>fast</td>
<td>1.11</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td>slow</td>
<td>1.14</td>
<td>2.5</td>
</tr>
<tr>
<td>Virtex-IIIE</td>
<td>fast</td>
<td>2.70</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>slow</td>
<td>2.70</td>
<td>1.1</td>
</tr>
<tr>
<td>Virtex-IIPro</td>
<td>fast</td>
<td>1.14</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>slow</td>
<td>1.10</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Analysis

Bridging Faults

- X, Y always driven with opposite polarity
  - Test Phase A: $X=0 \rightarrow 1$, $Y=1 \rightarrow 0$
  - Test Phase B: $X=1 \rightarrow 0$, $Y=0 \rightarrow 1$
- Bridge fault (any model, ex. wired AND/OR) causes X, Y lines to have same value
  - Causes $First_i$ to transition, $Last_i$ does not
  - $Clock_i$ transitions
  - Slice propagates $FST$ down chain
**Analysis**

**Bridging Fault Bus Line Interleaving**

- If set consists of more than 2 paths
  - Interleave X and Y lines on shared common bus: [Diagram]

**Fault Localization**

- No search iterations necessary
  - Configure chains of paths and logic blocks
  - Test
    - Incremental test result saved in each slice
      - Flip-flops save PST (pass) or FST (fail)
    - Scan out *(readback)* flip-flop values
    - XOR readback with expected value
      - First non-zero entry determines failing set
        - Quickly identifies set between 2 slices
Analysis
Fault Localization Example: Cfg. A

<table>
<thead>
<tr>
<th>readback file:</th>
<th>Q_{X_0}Q_{Y_0}Q_{X_1}Q_{Y_1}...Q_{X_{i-1}}Q_{Y_{i-1}}Q_{X_i}Q_{Y_i}Q_{X_{i+1}}Q_{Y_{i+1}}...Q_{X_N}Q_{Y_N}</th>
</tr>
</thead>
<tbody>
<tr>
<td>before test:</td>
<td>01 01 ... 01 01 01 01 ... 01</td>
</tr>
<tr>
<td>test passed:</td>
<td>10 10 ... 10 10 10 ... 10</td>
</tr>
<tr>
<td>test failed:</td>
<td>☒ 10 10 ... 10 11 11 ... 11</td>
</tr>
<tr>
<td></td>
<td>00 00 ... 00 01 01 ... 01</td>
</tr>
</tbody>
</table>

set_0  set_1 ... set_{i-1} set_i set_{i+1} ... set_N

determines failing set of paths as set_i

Outline

• Introduction
• Previous Work
• Implementation
• Analysis
• Conclusion
• Future Work
Conclusion

• FPGA interconnect delay fault testing method
  – Detects multiple small delay defects (~1-2 kΩ)
  – Also detects multiple
    • Stuck-at 0/1 faults
    • Stuck open faults
    • Bridging faults
  – Able to quickly localize faulty set of paths
  – Architecture-independent

Outline

• Introduction
• Previous Work
• Implementation
• Analysis
• Conclusion
• Future Work
Future Work

- Try out method on actual devices
  - Determine appropriate feedback delay values
  - Set detection sensitivity
- Will setup time violations cause problems? (no)
  - Defects between $t_{\text{defect}_{\text{max-pass}}}$ and $t_{\text{defect}_{\text{min-fail}}}$

References (1)


