Testing for Small-Delay Defects in Nanoscale Integrated Circuits

“I choose a block of marble and chop off everything that I do not need.”
Francoise Auguste Rodin (on how he created his statues)

Motivation: Decreasing Feature Sizes

- Process technology scales down continuously

Source: ITRS
Motivation: Increasing Defect Rates

- Sub-wavelength lithography (193 nm wavelength)

![Graph showing transistor length over time with labels for resistive opens, resistive shorts, and line edge roughness and residues.]

Motivation: Increasing Defect Rates

- Random dopant fluctuations in short transistor channel

![Diagram showing transistor operation with labels for depletion region, V<sub>d</sub> = 0 V, V<sub>C</sub> > V<sub>T</sub>, and V<sub>d</sub>.]

Transistor threshold voltage variations increase

3σ = 30%
Motivation: Increasing Defect Rates

- Closer wires

![Graph showing transistor length progression from 90nm to 45nm with increased crosstalk effect]

Motivation: Increasing Defect Rates

- Significant dynamic power density differences across the die

![Graph showing power dissipation with it causing voltage droop and power supply variations]

- Voltage droop
- Power supply variations
Motivation: Most Defects Cause SDDs

- All of these defects may cause small-delay defects (SDDs)
- Main contributors of SDDs
  - Transistor parameter variations
  - Power supply variations
  - Crosstalk
  - Resistive shorts and opens

Background

- High test-data volume and test-application times
  - Test-data volume 38x higher in 2015 than in 2007
  - Test application time 17x higher in 2015 than in 2007
- Many new types of defects cannot be accurately modeled using existing fault models.
  - Need to model the quality of test patterns such that they can be quickly assessed for defect screening
- Test selection is required to choose the most effective pattern sequences from large test sets (pattern grading)
  - Reorder patterns to reduce test time for abort-on-first-fail
  - Reduce pattern count for production test
  - Reduce CPU time for generating these patterns
- Common industry practice for test selection is based on fault grading
  - Computationally expensive
  - Must be repeated for every fault model
A Typical Motivating Scenario
(Acknowledgment: Phil Nigh, IBM)

- Semiconductor chip manufacturer needs to test 1 M copies of a chip with 10 K patterns (abort-on-fail)
- Typically only 2000 of the 10 K patterns are “unique fail patterns”
  - 70%-90% of production test patterns are useless (Ferhani et al., Stanford/IBM data, VTS 2008; Guo et al., Intel data, VTS 2006; Madge et al., LSI data, ITC 2004))
  - How do we tell which patterns to drop?
  - The 80/20 rule, The Vital Few versus The Trivial Many
  - Test economics challenge
- Majority of the fail patterns (e.g., 1800 out of 2000) occur in the first 5 K patterns
- Can we predict which 200 patterns of the next 5000 must be applied?
- Currently, all 5 K remaining patterns must be applied to get low DPM!

Shortcomings of Current Methods for Delay-Defect Test

- Stuck-at fault model alone not sufficient for high-quality test
- Traditional transition-test ATPG does not target small-delay defects (SDDs)
  - Inclined to select short activation paths
  - SDDs are observable on short-sack paths (long paths)
- Timing-aware ATPG tools have now emerged
  - Recent versions of Mentor Graphics FastScan, Cadence TrueTime ATPG, Synopsys TetraMax
  - Problems: High run times for large circuits, not addressing process variations, not layout-aware, over-reliance on static timing analysis and path enumeration
Short-Path Sensitization

Clock period: 6.5 ns
Small-delay defect size: 1 ns

Statistical Delay Quality Level (SDQL)

- Metric for test quality
- Assumes known delay-defect distribution $F(s)$
  Example: $F(s) = \lambda e^{-\lambda s}$
- $T_{mgn}$: slack on longest sensitizable path for this fault
  $T_{det}$: slack on actual propagation path

Example: $T_{clk} = 22$

$SDQL = \sum_{T_{mgn}}^{T_{det}} F(s) ds$ over all faults
Alternative Approach

• Strategy:
  – Use “output deviations” as a surrogate coverage-metric for test generation, pattern grading, seed selection for LBIST, and test compression
  – Use a flexible, but general, probabilistic fault model to generate a probability map for a logic circuit
  – Target multiple fault sites in a probabilistic manner
    • Compatible with existing test development flows
    • Useful for addressing phenomenon or mechanisms that are not fully understood
• Premise: Higher the deviation, better the quality of a test pattern
• Automation tools are being implemented for efficiently computing output deviations and grading test patterns for industrial circuits

IWLS 2005 Benchmarks

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Fault Count</th>
<th>Flip-Flop Count</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wb_dma</td>
<td>31,254</td>
<td>881</td>
<td>WISHBONE DMA/Bridge IP Core</td>
</tr>
<tr>
<td>tv80</td>
<td>40,022</td>
<td>359</td>
<td>TV80 8-Bit Microprocessor Core</td>
</tr>
<tr>
<td>systemcaes</td>
<td>52,544</td>
<td>670</td>
<td>SystemC AES</td>
</tr>
<tr>
<td>mem_ctrl</td>
<td>54,440</td>
<td>1,138</td>
<td>WISHBONE Memory Controller</td>
</tr>
<tr>
<td>usb_func</td>
<td>87,658</td>
<td>1,766</td>
<td>USB function core</td>
</tr>
<tr>
<td>ac97_ctrl</td>
<td>98,702</td>
<td>2,289</td>
<td>WISHBONE AC 97 Controller</td>
</tr>
<tr>
<td>aes_core</td>
<td>106,054</td>
<td>554</td>
<td>AES Cipher</td>
</tr>
<tr>
<td>dma</td>
<td>143,620</td>
<td>2,197</td>
<td>Direct Memory Access (DMA) Controller</td>
</tr>
<tr>
<td>pci_bridge</td>
<td>168,844</td>
<td>3,677</td>
<td>PCI Interface</td>
</tr>
<tr>
<td>wb_conmax</td>
<td>223,252</td>
<td>818</td>
<td>WISHBONE Conmax IP Core</td>
</tr>
<tr>
<td>ethernet</td>
<td>745,778</td>
<td>10,545</td>
<td>Ethernet IP core</td>
</tr>
<tr>
<td>vga_lcd</td>
<td>1,218,756</td>
<td>17,102</td>
<td>WISHBONE rev.B2 compliant Enhanced VGA/LCD Controller</td>
</tr>
<tr>
<td>netcard</td>
<td>6,647,092</td>
<td>97,381</td>
<td>Network Card Controller</td>
</tr>
<tr>
<td>leon3mp</td>
<td>6,896,250</td>
<td>108,839</td>
<td>32-bit processor compliant with SPARC V8 architecture</td>
</tr>
</tbody>
</table>
IWLS 2005 Benchmarks

- RTL models are available on IWLS website: http://www.iwls.org/iwls2005/benchmarks.html
- Netlist and layout-generation flow:

![Netlist and layout-generation flow diagram]

Challenge: ATPG for Small-Delay Defects

Timing-aware ATPG CPU time relative to TDF ATPG

![Relative CPU time and number of faults graph]

IWLS 2005 Benchmarks

Commercial ATPG tool
High Pattern Count
- Commercial timing-aware ATPG tools lead to large number of patterns

Limitations of Existing Methods (AMD Circuits)
- TA ATPG is expensive (compared to TDF ATPG)
**Research Need**

- Layout-aware and variation-aware pattern selection
  - Need to target real causes of SDDs
- Cost effective pattern selection for SDDs
  - Low CPU time
  - Small pattern count
- Effective SDD detection
  - Cover all high risk paths

---

**Test-Pattern Grading and Pattern Selection**

- Gate Delay Defect Probabilities (DDP)
  - Gate delay has a distribution
  - DDP: Probability that the delay of a gate is larger than a delay limit
    - Set a critical delay limit for the gate: $D_{crt}$ (Relaxed limit, e.g., max delay from STA)
    - If delay is above $D_{crt}$: Delay defect
    - DDP: Probability that the gate delay is more than $D_{crt}$ for the given input transition

[Diagram showing probability distributions and delay defects]
Test-Pattern Grading and Pattern Selection

- Delay Defect Probability Matrix (DDPM)
  - Includes DDPs for all input $\rightarrow$ output timing arcs
  - Example: DDPM for an OR2 gate (entries are arbitrary)

- Signal Transition Probabilities (STPs)
  - Delay-fault test-patterns will force signal transitions on circuit nets
  - 4 different signal transitions are possible:
    - Low $\rightarrow$ Low, Low $\rightarrow$ High, High $\rightarrow$ Low, High $\rightarrow$ High
    - Each of these events has a probability to occur
    - Each net has a vector of signal-transition probabilities:
**Example**

\[ \begin{align*}
\text{A} & \quad \text{< 0.2, 0.8, 0, 0 >} & \quad \text{< 0.4, 0.6, 0, 0 >} & \quad \text{< 0.616, 0.384, 0, 0 >} \\
\text{B} & \quad \text{< 0.2, 0.8, 0, 0 >} & \quad \text{< 0.4, 0.6, 0, 0 >} & \quad \text{< 0.616, 0.384, 0, 0 >} \\
\text{Z} & \end{align*} \]

- Both A and B stay @ LOW \(\rightarrow\) No delay defect activated: \[0.2 \times 0.4 = 0.08\] (Z stays @ LOW, no defect)
- A or B stays @ LOW and the other input switches \(\rightarrow\) No delay defect activated: \[0.2 \times 0.6 + 0.4 \times 0.8 = 0.44\] (Z stays @ LOW, no defect)
- Both A and B make LOW \(\rightarrow\) HIGH transition:
  - Delay-defect: \[0.8 \times 0.6 \times 0.2 = 0.096\] (Z stays @ LOW, defect case)
  - No defect: \[0.8 \times 0.6 \times (1 - 0.2) = 0.384\] (Z goes LOW \(\rightarrow\) HIGH, no defect)
- Overall output STP: \(Z = < 0.08 + 0.44 + 0.096, 0.384, 0, 0 > = < 0.616, 0.384, 0, 0 \)

---

**Test-Pattern Grading and Pattern Selection**

- An example (arbitrary DDPMs):

Initialization of signal transition probabilities on INs

Expected signal transitions are shown in dark boxes
Probabilistic Delay-Fault Model and Output Deviations

- Propagation of Signal-transition probabilities (STPs)
  - The nets connected to the test-application points: Initialization nets (INs) → Initialized with “0” DDP
  - During signal propagation through circuit, use DDPM of the gates to update signal-transition probabilities
    - Net: \( <P_{L \rightarrow H}, P_{L \rightarrow L}, P_{H \rightarrow L}, P_{H \rightarrow H}> \)

  The probability that net \( A \) will have expected signal-transition, \( P_{\text{EXPECTED}} \)
  Deviation: \( 1 - P_{\text{EXPECTED}} \)

Test-Pattern Grading and Pattern Selection

- An example (arbitrary DDPMs):

  There is no transition on net \( E \).
  The probability of a delay fault (deviation) is 0.
Test-Pattern Grading and Pattern Selection

- An example (arbitrary DDPMs):

<table>
<thead>
<tr>
<th>XOR2</th>
<th>Initial Input State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00</td>
</tr>
<tr>
<td>Inputs</td>
<td>IN0</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
</tr>
</tbody>
</table>

Inputs:

- IN0: 0.3, 0.4, 0.2, 0.3
- IN1: 0.3, 0.4, 0.1, 0.4

The output changes due to IN1. Probability of a delay fault: 0.4

Test-Pattern Grading and Pattern Selection

- Output deviation
  - The probability that the output value is different from the expected value
  - Relative deviations at the observation points are considered
  - For the applied test pattern, Q2 is more prone to SDDs

Output Deviations:

- Q1: 0.52
- Q2: 0.664
Example

The output deviation (for each observable output) for an input pattern is the probability that the output value is different from the expected value.

Output Deviations:
Q1: 0.52
Q2: 0.664

– Linear-time computation (ignore signal correlations, reconvergent fanout)
– No need to enumerate paths

Probabilistic Delay-Fault Model and Output Deviations

• Rules of STP Propagation
  1) If output does not change, the deviation on output net is 0.
  2) If any one of the multiple input-transitions can cause the output transition, only the maximum deviation provider is considered.
  3) If multiple input-transitions are required for an output transition, all required input-transitions are considered.

• Deviation always increases through a sensitized path (formal proof)
Test-Pattern Grading and Pattern Selection

- Pattern selection method
  - For each pattern, calculate output deviations for all outputs
  - Drop ineffective patterns on the fly (apply a lower deviation threshold)
  - For each output, keep a list of most effective patterns

- Final pattern ordering
  - The patterns effective for most outputs come first
  - The goal is to quickly increase topological coverage

Pattern Selection

Patterns are ordered according to the deviation that they caused at the corresponding observation point.

We will select 3 patterns.
Pattern Selection

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>P2</td>
<td>P2</td>
</tr>
<tr>
<td>P2</td>
<td>P5</td>
<td>P9</td>
</tr>
<tr>
<td>P3</td>
<td>P5</td>
<td>P7</td>
</tr>
<tr>
<td>P5</td>
<td>P7</td>
<td>P6</td>
</tr>
</tbody>
</table>

Start with the first observation point. Select P1.

Selected Patterns

- P1

Pattern Selection

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>P5</td>
<td>P9</td>
</tr>
<tr>
<td>P3</td>
<td>P5</td>
<td>P7</td>
</tr>
<tr>
<td>P5</td>
<td>P7</td>
<td>P6</td>
</tr>
</tbody>
</table>

Continue with Q2. Select P2.

Selected Patterns

- P1
- P2
Pattern Selection

Continue with Q3. P1 has already selected. Select P9.

Selected Patterns

- P1
- P2
- P9

Simulation Results

- Generating DDPMs
  - HSpice Monte Carlo (MC) simulations
    - 200 MC simulations
    - Transistor gate length, L, 3σ = 10%
    - Threshold voltage, Vth, 3σ = 30%
    - Gate-oxide thickness, tox, 3σ = 3%
Simulation Results

• Correlation Between Output Deviations and Path Lengths
  – Kendall’s correlation coefficient
  – Perfect positive correlation measure: 1
  – Near-perfect correlation between output deviations and path lengths!

![Bar chart showing Kendall's correlation coefficients for various benchmarks.]

Simulation Results

• Benchmarks
  – IWLS 2005 benchmarks → Synthesized ASIC blocks
• Simulation setup
  – Commercial ATPG tool
    • n-detect TDF ATPG patterns
    • TA ATPG patterns
  – Servers
    • Minimum 16 GB RAM
    • Quad-core Opterons
  – In-house tools
    • Coded in C++

• Comparison Points
  – Dynamic-timing simulation
    [Lee et al, DFT Symp. 2006]
    • Approximate path delays are calculated instead of output deviations
    • Always selects top 1/3 of patterns
  – Timing-aware ATPG
    • Commercial ATPG tool used to generated timing-aware ATPG patterns
Simulation Results

- CPU time usage

Simulation Results

- Long path coverage ramp-up (systemcaes)
Simulation Results

- Delay-defect injection experiments
  - Fault coverage ramp-up (usb_funct)

Layout-aware Output Deviations

- The method of output deviations has been enhanced to target interconnects

- Interconnects play a major role in circuit delays and delay variations [ITRS 2007]
  - Crosstalk
  - Process variations on wire geometries

- Interconnect- and layout-awareness is required for realistic results
Layout-Aware Output Deviations

- Assign a buffer-like DDPM to all wires
- How detailed do we need to model?
  - **Lumped delay model**: Single DDPM for all connected wires (for each net)
    - Not accurate enough
    - May lead to misleading results

Other options:

- **Pin to pin delay model**:
  - More accurate
  - What if we need more resolution?

- **Via to Via delay model**:
  Model each metal layer separately
  - Run time penalty: Is it worth it?
Layout-Aware Output Deviations

- STP propagation (pin-to-pin delay model)

```
<1,0,0,0>  <1,0,0,0>  <1,0,0,0>  <0,6,0,0,0>  <0.7,0,0,0>
0->0  0->0  0->0  0->0  0->1
<0,1,0,0>  <0,1,0,0>  <0.1,0,9,0,0>  <1,0,0,0>  0.7,0,0,0>
0->0  0->0  0->1  0->1  0->1
<0,0,0,1>  <0,0,0,1>  <0.35,0,65,0,0>  <0,4,0,6,0,0>  <0.45,0,65,0,0>
1->1  1->1  0->1  0->1  0->1
<0,1,0,0>  <0,1,0,0>  <0.3,0,7,0,0>  <0.4,0,6,0,0>  0.45,0,65,0,0>
0->0  0->0  0->1  0->1  0->1
```

Output deviation = 0.7

Output deviation = 0.45

---

Results

- Normalized CPU time (normalized by TA ATPG data)

```
<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>total(n=3)</th>
<th>total(n=5)</th>
<th>total(n=8)</th>
<th>total(n=10)</th>
<th>li-aware</th>
</tr>
</thead>
<tbody>
<tr>
<td>wb_dma</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tv60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>systemscaes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mem_ctrl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>usb FUNCT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ac97_ctl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aes_core</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dmva</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pc_bridge</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wb_conmax</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Normalized CPU time
Results

- Breakdown of CPU time

- Fault coverage ramp-up (tv80)
Summary of Simulation Results

- The effect of considering interconnect delays
  - 15-40% more excited long paths
  - Less than 15% CPU time penalty when pin-to-pin wire delay model is used
  - Significant advantages at low run-time cost

Results for Industry Circuits

- Designs
  - Four different AMD circuit blocks
  - Blocks are selected from different functional units

<table>
<thead>
<tr>
<th>Design</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit A</td>
<td>Cache related</td>
</tr>
<tr>
<td>Circuit B</td>
<td>In execution unit</td>
</tr>
<tr>
<td>Circuit C</td>
<td>In execution unit</td>
</tr>
<tr>
<td>Circuit D</td>
<td>In load-store unit</td>
</tr>
</tbody>
</table>

- ATPG
  - Commercial ATPG tool
  - n-detect TDF ATPG and TA ATPG patterns

- Simulation environment
  - Pool of servers (10000+) with at least 16GB of free RAM
  - Simulation programs were coded in C++
Experimental Results

- Normalized number of sensitized long paths (Long path limit = 70% CLK)

Results for AMD Circuits

- Normalized number of test patterns (Long path limit = 70% CLK)
Results for AMD Circuits

- Normalized CPU time usage (Long path limit = 80% CLK)

Simulation Results

- Long path coverage ramp-up (Long path limit = 80% CLK, Circuit A)
Conclusions

• Rethinking of the ATPG problem for small-delay defects
• Output deviations provide an effective metric for pattern grading
• Test-pattern selection for small-delay defects at the gate level
  – Reduce the need for timing-aware ATPG
  – Low pattern count and less CPU time