Coverage Metrics for Post-Silicon Validation

Tim Cheng, Ming Gao, and Peter Lisherness
Univ. of California, Santa Barbara
Coverage Metrics for Post-Si Validation

PI: Cheng
Inadequacy of Existing Metrics for Post-Si Validation

Structural metrics inadequate
- High computational complexity
- Not meaningful for implementation errors
- Risk of over-testing

Functional metrics inadequate
- Focus primarily on activation
- No observability measure
- Do not directly address electrical bugs
What Is Needed?

Measure of observability
- Ignored by many functional metrics
- Accessibility limited to chip IOs, DfX

High-level compatibility
- Efficient large-scale simulation
- Support verification functional models
- Support HLS design

Low-level awareness
- Target electrical bugs (automatically and semi-manually)
Tests cover “bins”. Coverage is necessary but not sufficient for detecting bugs!
Coarse metrics (e.g. toggle) are easy to compute but saturate early and can lead to over-confidence.
Highly granular metrics (e.g. path coverage) give a much better picture, but are rarely scalable.
N-detection sets a multiple cover threshold, making bins harder to cover, ideally increasing bug coverage.
An Infinite Error Space

White boxes (simulation) allow checkers everywhere, black boxes (silicon) do not and limit test efficacy.

- All registers are fully observable (white-box assumption)
- Only system-outputs are observable (black-box assumption)
Two Recently Completed HL Coverage Tools

**Observability Coverage (HLDVT 2009)**
- Symbolic (model-free) coverage measure
- Extension of OCCOM [Fallah DAC’98] to higher levels
- Implemented with GCC

**Mutation Coverage (DAC 2010)**
- Design error injection coverage measure
- Well established in software testing
- Also implemented with GCC
Integrating with GCC

GCC: The GNU Compiler Collection

Front-end: Language specific
Middle: Instrumentation and Optimization Passes
Back-end: Target Code Generation

Source Code: C C++ SystemC etc.

Executable Binaries: x86 amd64 ia64 etc.

• Plug-in custom pass
• Perform static analysis
• Add coverage instrumentation or inject error
Observability Coverage

- Hypothetical error symbols
- Unique tag for every, or targeted, expression
- Tags covered when they reach a PO / observation point

Design errors as ‘tags’

\[ b = 42 + \Delta \]

\[ ... \text{c=False} \]

\[ \text{if (c) } b = 0 \]

\[ a = b + 5 \quad (a = 47 + \Delta) \]

output(a)
Our Approach: Tag Instrumentation (HLDVT 2009)

- Attach tags to data
- Track during simulation
- Use shadow variables: “trackers”

```
metadata Tracking

b=42 +Δ₁
Tr_b = Δ₁
...

if (c) b=0 +Δ₂
  Tr_b = Δ₂
a=b+5 +Δ₃
Tr_a = Tr_b | Δ₃
output(a)
_observe(Tr_a)
```
Branch Handling

- **Data-control dependencies**
  - E.g.: if-else Mux
  - Tag approximation

- **Use dependence stack**
  - Push tracker at branch
  - Pop at immediate post-dominator
  - Loop safe
  - From [Xin ISSTA’07]

```plaintext
a = get()  + Δ
...
if (a)
  b = c  + Δ
else
  b = 0  + Δ
output(b)
```
SCEMIT – A SystemC Error and Mutation Injection Tool (DAC 2010)

**High compatibility**

- Uses GCC plug-in interface
- Supports C, C++, and SystemC
- Minimal simulation flow changes needed to compute

**Abridged model**

- Operates on GCC internal AST
- Only a subset of mutation operators make sense
SCEMIT also allows targeting of any mutant from other sources

- Mutants extracted from verification bins
- Mutants mapped from low-level simulation results
- ...

### Table: Mutant/Error Types Automatically Generated in SCEMIT

<table>
<thead>
<tr>
<th>Name</th>
<th>Full Name</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPR</td>
<td>Operator Replacement</td>
<td>$a=b+1 \Rightarrow a=b-1$</td>
</tr>
<tr>
<td>VCR</td>
<td>Var$\Rightarrow$Constant Repl.</td>
<td>$\text{if (a)} \Rightarrow \text{if (true)}$</td>
</tr>
<tr>
<td>CCR</td>
<td>Constant Replacement</td>
<td>$a=b+1 \Rightarrow a=b+0$</td>
</tr>
<tr>
<td>ROR</td>
<td>Relational Op. Replacement</td>
<td>$a\geq b \Rightarrow a==b$</td>
</tr>
</tbody>
</table>
Error Injection Methods

a) Original

b) Source-to-source
   - Used by mutation tools (Certify, ProteumIM)

c) Modified runtime
   - Used by C++/SystemC error injection tools

d) Compiler injection
   - Used in SCEMIT
Incorporating Activation Criteria into Observability Coverage

Enhance beyond propagation-only

- Need activation criteria beyond ‘executed’
- Attach tags/mutants to verification bins
  - Based on human knowledge; Allows for targeting

Target electrical bugs

- Hybrid model
  - A tag/mutant is activated only if certain LL activation conditions are met
  - Propagate activated tags/mutants at high-level
- Use N-detection
  - No tag/mutant dropping before detecting it N times
  - Increasing chance of detecting electrical bugs
Fault/Error Models for Electrical Bugs: Combining LL Accuracy and HL Efficiency

Ideal Model:
1. Accurately reflect electrical bug activation & propagation
2. Affordable at full-chip level

- C/SystemC
- RTL HDL
- Gate Level
- SPICE Model
- Layout
- Silicon

Delay Fault
Parameter Variation (e.g. Vdd, Vth)
Characterization: Parasitic Extraction, Victim Assessing, etc.
SI noise sources: xtalk, power droop, etc.
Design Marginality
Process Variation

GAP
More Scalable
More Accurate
A fault is considered activated:

1) Transition: aggressor/victim, and
2) A critical path sensitized

A critical path sensitized

1) Fault site: Critical PPO
2) Activation Constraint: transitions on critical nodes or noise sources
HELM: Hybrid Electrical Bug Model

**Constrained Activation**

**Bit-Flip**

**N Detection**

**Observability Evaluation**

**Critical PPO**

Constraints (critical nodes, segments, etc.)

HELM Fault Activation

Error Injection

Error Propagation Evaluation

DfX

System Output
HELM Modeling Flow

**INPUT**
- Post-Si Functional Test Suite
- Timing/Noise Analysis

**OUTPUT**
- N-Detection Fault Coverage Statistic
- Adv. Reports
  - Test Selection, DfX Guidance, Cov. Closure, etc.

**PROCESS**
- HELM Fault Propagation: HL Error Injection and Simulation
- LL-to-HL Error Mapping
- HELM Error Injection Log: "<Cycle>, <PPO>"
- A list of activated HELM faults:
- Fault Activation Recorder: Gate-Level Logic Simulation

**HELM Fault Propagation:**
- HL Error Injection and Simulation
- LL-to-HL Error Mapping
- HELM Error Injection Log: "<Cycle>, <PPO>"
- A list of activated HELM faults:
- Fault Activation Recorder: Gate-Level Logic Simulation

**Fault Activation Recorder:**
- A list of modeled faults: "<a critical PPO + constraints>"
Case Study: An RTL Implementation of Alpha 21264

- Consider entire processor but focus on 3 pipeline stages (decode, simple_ALU, and branch) for bug modeling/detection.

- Generated 1000 random chip instances with different timing characteristics, mimicking fabricated chips -- resulting in some good, and some buggy chips.

- Generated 100 pseudo-functional test programs
  - filling cache memory with random instruction sequences
  - each program was executed for 100 clock cycles

- Evaluate and compare quality of selected tests based on:
  - HELM
  - DSIM (golden, near-optimal)
  - TGL (baseline, commonly used in practice)
Specific HELM Model Chosen for Test Selection in Case Study

**HELM Fault Activation:**
- Activated if required transitions at selected critical PPO-PPI pair at same cycle.

**HELM Fault Detection:**
- Bit-flip injected in RTL variable mapped from activated PPO with cycle stamp.
- Detected if error propagate to obsv. point.

**Progressive N-detection based test selection:**
- Prioritize covering as many faults as possible than covering each of them as many times before reaching the limit N.

```plaintext
......
// error: B[0] @ 12
Y = B + X;
......
//obsv. point
output(Y);
```
Test Selection Quality: HELM Better Than Toggle

Coverage of Defective Instances

<table>
<thead>
<tr>
<th>M</th>
<th>DSIM</th>
<th>HELM</th>
<th>TGL</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>83%</td>
<td>61%</td>
<td>37%</td>
</tr>
<tr>
<td>10</td>
<td>94%</td>
<td>66%</td>
<td>43%</td>
</tr>
<tr>
<td>20</td>
<td>100%</td>
<td>77%</td>
<td>62%</td>
</tr>
<tr>
<td>30</td>
<td>100%</td>
<td>81%</td>
<td>67%</td>
</tr>
</tbody>
</table>
Model Accuracy Loss due to Absence of Error Propagation Evaluation

Coverage of Defective Instances

- DSIM
- HELM
- HELM-A

Selected Top-M tests

Coverage of Defective Instances

0.0% 10.0% 20.0% 30.0% 40.0% 50.0% 60.0% 70.0% 80.0% 90.0% 100.0%
0 10 20 30 40 50 60 70 80 90 100

Selected Top-M tests
Error Propagation: A Big Deal

Count of Defective Instances Killed by Top M Tests

Selected Top-M tests

Count of Defective Instances Killed by Top M Tests

DSIM-A

DSIM

565

234

0

100

200

300

400

500

600

0 10 20 30 40 50 60 70 80 90 100

Selected Top-M tests
Activation Constraints Contribute to Model Accuracy

Coverage of Defective Instances

- **DSIM-A**
- **TGL**
- **HELM-A**

<table>
<thead>
<tr>
<th>Cov.</th>
<th>DSIM-A</th>
<th>HELM-A</th>
<th>TGL</th>
</tr>
</thead>
<tbody>
<tr>
<td>75%</td>
<td>M=4</td>
<td>M=16</td>
<td>M=31</td>
</tr>
<tr>
<td>80%</td>
<td>M=5</td>
<td>M=22</td>
<td>M=40</td>
</tr>
<tr>
<td>85%</td>
<td>M=7</td>
<td>M=24</td>
<td>M=64</td>
</tr>
<tr>
<td>90%</td>
<td>M=10</td>
<td>M=37</td>
<td>M=66</td>
</tr>
<tr>
<td>95%</td>
<td>M=17</td>
<td>M=44</td>
<td>M=76</td>
</tr>
</tbody>
</table>

Selected Top-M tests
Scalability and Accuracy Tradeoff

Coverage of Defective Instances

- DSIM
- HELM
- HELM-10%

Selected Top-M tests

Coverage of Defective Instances

<table>
<thead>
<tr>
<th># of Injections</th>
<th>Sim. Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>HELM</td>
<td>481,514</td>
</tr>
<tr>
<td>HELM-10%</td>
<td>48,152</td>
</tr>
</tbody>
</table>
• HELM: a Hybrid ELectrical bug Model
  • Low-level analysis to select targets and their activation criteria
  • Single-pass RTL simulation to derive all activated faults
  • High-level simulation to efficiently compute the propagation and observability for each activated fault

• Experimental results show:
  • Observability is crucial for bug detection
  • N-detection of HELM faults help improve bug coverage
  • Tests selected based on HELM detect significantly more buggy instances than those selected based on toggle coverage

• HELM should be useful for guiding DfD
Backup slides
Observed Coverage Experiment

Tested on T48 (i8048) microcontroller core ISS

- Available from opencores.org
- Modified ISS for experiment:
  - All port and bus outputs as POs
  - Halt after 1,000 cycles; “Scan out” at end of simulation

Results compared to statement and mutation coverage

- Tags injected into i8039.c
- Statement coverage from GCC’s gcov tool
- Mutation coverage from the ProteumIM C mutation tool [Delmaro]
Results

Cumulative Coverage for Random Test Campaign

- Absolute values not directly comparable
  - 3 different coverpoint sets
- Feasibility gaps
  - Predication conditions
  - Equivalent mutants

Statement (5s)
Mutation (19m22s)
Observability w/predication (4m 52s)
Observability (1m4s)
Saturation means metric stops working
- Want later saturation
- Uncovered points most important
- Indicates test inadequacy
Results – Normalized Functional

Cumulative Coverage for Functional Test Campaign - Normalized

- Functional test set did not saturate any metric
- Better agreement between metrics
  - Expected, since functional tests try to propagate activated behavior

Statement (4s)  Mutation (79m51s)
Observability w/predication (1m 28s)  Observability (59s)
Observability Coverage Summary

### Execution Time

<table>
<thead>
<tr>
<th>Test Set</th>
<th>No. of Test Progs.</th>
<th>Cycles Simulated</th>
<th>Execution Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Statement</td>
</tr>
<tr>
<td>Random</td>
<td>100</td>
<td>100,000</td>
<td>5</td>
</tr>
<tr>
<td>Functional</td>
<td>91</td>
<td>17,515</td>
<td>4</td>
</tr>
</tbody>
</table>

Consistent with other observability metrics

- Better than statement coverage - slower to saturate
- Faster than mutation
  - Smaller coverpoint population (46000 mutants vs. 2000 tags)
- Basic instrumentation overhead is negligible
Adding N-Detection for Granularity and Performance

- Heuristic employed in manufacturing testing
  - Every single-stuck-at/transition detected N times
- HELM selection granularity
  - Each constrained critical PPO toggle only has statistical chance to activate electrical bugs
  - Require N toggles to improve likelihood of catching error
- HELM simulation efficiency
  - Drop HELM fault after N detections