Measuring the Effectiveness of Verification Environments

Yogish Sekhar
Verification Specialist
Dialog Semiconductor
Can We Really Measure Verification Effectiveness

- Yes We Can
- Orthogonal To Traditional Techniques
- Improves
  - Confidence
  - Quality
- Reduces Overall Time For Closure
Complexities In Today’s Designs

- Domain Dependency
- Standards
- Legacy Designs
- Backward Compatibility
- Reusability
- Configurability
- Development Process
- Methodologies And Tools
- Time To Market
Why Measure Verification Effectiveness?

- Verification Effectiveness ➔ Bug catching capabilities
- If there was a bug in the design - can the VE find it?
- "To measure is to know" - Lord Kelvin
Bug Classification

- Global Error Scenarios
  - Failure to understand a requirement
  - Failure to satisfy specifications
  - Failure to write specifications

- Design Specific Error Scenarios
  - Failure to satisfy a requirement
  - Missing control paths
  - Inappropriate path selection
  - Inappropriate or missing actions
Mechanics of Bug Detection

**PHASE 1**

ACTIVATE

Simulate the logic that contains the bug

**PHASE 2**

PROPAGATE

Propagate the bug to observable points

**PHASE 3**

DETECT

Detect the bug (Checkers/Assertions)
Inject Artificial Bugs

1. Stimulus
2. Design
3. Compare

Reference Model

Verification Environment

Tests Pass

Tests Fail
## Fault Classes and Types

<table>
<thead>
<tr>
<th>Fault Types</th>
<th>CLASS Where the fault is injected</th>
<th>TYPE What type of fault is injected</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Port Stuck at 0/1/value</td>
<td>Port Negated</td>
</tr>
<tr>
<td><strong>Fault Classes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TopOutputs Connectivity</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>ResetConditionTrue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SynchronousControl Flow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>InternalConnectivity</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>SynchronousDead Assign</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ComboLogicControl Flow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SynchronousLogic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ComboLogic</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
UVM Based - Block Level Qualification – Initial Results

- Qualification started when
  - Expression Coverage ~ 95%
  - Functional Coverage ~ 90%
- Power Sequencer Design Code ~ 3000 Lines
- 14/5000 Constrained Random Tests Used
UVM Based - Block Level Qualification – Final Results

- Potential Error Scenarios
  - 95% Detected / Reviewed

- Overall Reduction of 2 Man Weeks Effort In Closing Block Verification

Fault classes for 'pwr_on_off_ANY_00'

This report was generated on: 2013-02-19 at 10:55:22

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Faults In Design</th>
<th>Faults In List</th>
<th>Non-Activated</th>
<th>Activation Ratio</th>
<th>Non-Propagated</th>
<th>Detected</th>
<th>Non-Detected</th>
<th>Disabled By Certitude</th>
<th>Disabled By User</th>
<th>Dropped</th>
<th>Not Yet Qualified</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectivityOutput</td>
<td>144</td>
<td>144</td>
<td>0</td>
<td>100.00 %</td>
<td>0</td>
<td>130</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ResetConditionTrue</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td>100.00 %</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SynchronousControlFlow</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>100.00 %</td>
<td>0</td>
<td>81</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ConnectivityInput</td>
<td>1389</td>
<td>1241</td>
<td>0</td>
<td>100.00 %</td>
<td>9</td>
<td>910</td>
<td>0</td>
<td>0</td>
<td>312</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>SynchronousDeadAssign</td>
<td>13</td>
<td>13</td>
<td>0</td>
<td>100.00 %</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ComboLogicControlFlow</td>
<td>179</td>
<td>179</td>
<td>0</td>
<td>97.58 %</td>
<td>8</td>
<td>153</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>SynchronousLogic</td>
<td>61</td>
<td>61</td>
<td>0</td>
<td>98.25 %</td>
<td>1</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>ComboLogic</td>
<td>320</td>
<td>320</td>
<td>0</td>
<td>93.61 %</td>
<td>22</td>
<td>197</td>
<td>0</td>
<td>0</td>
<td>54</td>
<td>47</td>
<td>0</td>
</tr>
<tr>
<td>OtherFaults</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>N/A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>All Fault Classes (9)</td>
<td>2257</td>
<td>2057</td>
<td>0</td>
<td>N/A</td>
<td>40</td>
<td>1545</td>
<td>0</td>
<td>0</td>
<td>409</td>
<td>63</td>
<td>0</td>
</tr>
</tbody>
</table>
Full Chip Based – Block Level Qualification

- Charger: 100% Explained Coverage
- System FSM: 100% Explained Coverage
- SPI: 100% Explained Coverage
- Vibrator: 100% Explained Coverage
- Bist ADC: 100% Explained Coverage
Limitations

- Works On Static Mutations Of RTL
- No Ability To Introduce Timing Related Faults
- Increases Demand For Resources & Licenses
- Testcase Runtimes Have Impact On Performance
- Analysing Non-Propagated and Non-Activated Mutations Take Time
Recommendations

- Every Non-Detected Mutation Is A Potential Bug
  - Additional checks are needed to improve bug detection capability.

- Analyse Every Non-Propagated Mutation For Its Merit
  - Some of these mutations may need additional configurations/stimulus
  - Others may not have any functional impact.

- Achieving 100% Activation Score Is tough
  - Certain mutations that will never be activated
  - Mutations may be present in a redundant part of the design
  - Analyse the design to see if the code can be removed.
Recommendations

- Licence & Resource Usage
  - Mutation analysis tools are heavy users
- Use Short Test cases
  - Long test cases have significant impact on performance
- Constrained Random tests Can Be Used
  - Use explicit seed values
- Re-entrant Test cases
  - Parallel processing.
  - Run multiple tests for a single mutation
  - Run multiple mutations using a single testcase at any time.
  - Else test cases are serialised => impact performance significantly
- Checker Architecture
  - Construct Checkers/Scoreboards that use only inputs to create expected output.
Conclusions

- “If You Can’t Measure; You Can’t Manage”

- Measuring Verification Environment Effectiveness
  - Objective view on
    - Bug Catching Capability
    - Stimulus
    - Quality
  - Identify Potential Environment Weak Spots
  - Increased Chances Of Catching Real Bugs
  - Easy To Integrate Into Existing Verification Framework
Acknowledgements

- Saima Amer – Dialog Semiconductor
- Stuart Levine – Dialog Semiconductor
- Ajay Sharma - Synopsys
The power to be...