Improving Simulation Regression Efficiency using a Machine Learning-based Method in Design Verification

Deepak Narayan Gadde, Infineon Technologies Dresden GmbH & Co. KG
Sebastian Simon, Infineon Technologies Dresden GmbH & Co. KG
Djones Lettnin, Infineon Technologies AG
Thomas Ziller, Cadence Design Systems GmbH
Agenda

• Motivation and Problem Statement
• Cadence Xcelium ML
• Application on Designs
• Xcelium ML vs Ranking
• Proposed Methodology
• Summary
Agenda

• Motivation and Problem Statement
  • Cadence Xcelium ML
  • Application on Designs
  • Xcelium ML vs Ranking
  • Proposed Methodology
  • Summary
Verification of Systems on Chip became a time-consuming task

- Designs are becoming more complex, due to
  - Technology scaling: Ability to fabricate more
  - Consumer demands more from a single chip
  - Mixed-signal designs
  - Safety/Security critical devices and more...

- At present, pre-silicon verification typically consumes significantly more effort than design
Money and time are crucial...

- **Problem:** Huge number of test simulations in the Regression
  - High demand for simulator licenses
  - Long turn-around times
  - Uncertainty in coverage regain

- **Goals:**
Common Solution used so far - Ranking

**Definition:** A collection of test-seed pairs of the original regression to reach its achieved coverage

- **Pro:**
  - Easy
  - built-in (e.g. vManager)
  - Proven technology

- **Con:**
  - Effectively re-simulation (like directed testing)
  - Exact same coverage as original regression
  - Inability to hit new coverage/ identify new bugs

---

### Pro:
- Easy
- built-in (e.g. vManager)
- Proven technology

### Con:
- Effectively re-simulation (like directed testing)
- Exact same coverage as original regression
- Inability to hit new coverage/ identify new bugs
Agenda

• Motivation and Problem Statement
• Cadence Xcelium ML
  • Application on Designs
  • Xcelium ML vs Ranking
  • Proposed Methodology
• Summary
Cadence Xcelium ML High Level View

Xcelium™ ML analyzes patterns hidden in verification regression results

Source: Cadence Design Systems, Inc
Faster Regressions with Xcelium ML

Original Regression
- 50 tests, 100 seeds per test (5,000 runs)

Random control

```plaintext
class cfg_c extends uvm_sequence_item;
rand focus_e focus;
rand [$t8] rank;
...
endclass

function void test::setup();
cfg_c cfg = get_config();
cfg.randomize();
set_config_info(cfg);
endfunction
```

Generated Regression
30 tests, 1,500 runs

Xcelium™ ML Regression Coverage model

Some bins not regained

Some bins newly hit

Generate new regression runs

Original Regression Coverage model
Agenda

• Motivation and Problem Statement
• Cadence Xcelium ML
• Application on Designs
• Xcelium ML vs Ranking
• Proposed Methodology
• Summary
Application on an SoC Design - Overview

• Mixed Signal SoC
• Designed in VHDL, Verilog, SystemVerilog
• UVM (SV) testbench
• 356 Distinct Tests
• ≈ 5124 runs in the Regression

<table>
<thead>
<tr>
<th>Configuration</th>
<th>CPU</th>
<th>Verification IP</th>
<th>Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Yes</td>
<td>No</td>
<td>Simple</td>
</tr>
<tr>
<td>P2</td>
<td>Yes</td>
<td>Yes</td>
<td>Complex</td>
</tr>
<tr>
<td>P3</td>
<td>Yes</td>
<td>Yes</td>
<td>Simple</td>
</tr>
<tr>
<td>P4</td>
<td>No</td>
<td>No</td>
<td>Complex</td>
</tr>
<tr>
<td>P5</td>
<td>No</td>
<td>No</td>
<td>Simple</td>
</tr>
<tr>
<td>P6</td>
<td>No</td>
<td>Yes</td>
<td>Complex</td>
</tr>
<tr>
<td>P7</td>
<td>No</td>
<td>Yes</td>
<td>Simple</td>
</tr>
</tbody>
</table>
## Application on an SoC Design – Detailed Results

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Original Runs</th>
<th>Original Runtime CPU hours</th>
<th>Optimization Runs</th>
<th>Opt. Runtime CPU hours</th>
<th>Compression in Runs</th>
<th>Coverage Regain</th>
<th>Compression in Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>297</td>
<td>0.200</td>
<td>110</td>
<td>0.151</td>
<td>2.70</td>
<td>99.86%</td>
<td>1.32</td>
</tr>
<tr>
<td>P2</td>
<td>415</td>
<td>0.562</td>
<td>188</td>
<td>0.337</td>
<td>2.20</td>
<td>101.2%</td>
<td>1.66</td>
</tr>
<tr>
<td>P3</td>
<td>2959</td>
<td>2.813</td>
<td>786</td>
<td>0.785</td>
<td>3.76</td>
<td>97.96%</td>
<td>3.58</td>
</tr>
<tr>
<td>P4</td>
<td>140</td>
<td>0.213</td>
<td>116</td>
<td>0.133</td>
<td>1.20</td>
<td>98.07%</td>
<td>1.60</td>
</tr>
<tr>
<td>P5</td>
<td>1193</td>
<td>1.336</td>
<td>318</td>
<td>0.356</td>
<td>3.75</td>
<td>99.48%</td>
<td>3.75</td>
</tr>
<tr>
<td>P6</td>
<td>10</td>
<td>0.007</td>
<td>3</td>
<td>0.003</td>
<td>3.33</td>
<td>98.19%</td>
<td>2.33</td>
</tr>
<tr>
<td>P7</td>
<td>110</td>
<td>0.074</td>
<td>84</td>
<td>0.070</td>
<td>1.30</td>
<td>101.3%</td>
<td>1.05</td>
</tr>
<tr>
<td>Total</td>
<td>5124</td>
<td>5.205</td>
<td>1605</td>
<td>1.835</td>
<td>3.19</td>
<td>99.42%</td>
<td>2.83</td>
</tr>
</tbody>
</table>
Application on other Designs

**Microprocessor Digital IP**
- VHDL design
- UVM-SV testbench
- Already taped out
- 260 runs with 26 distinct tests

- Results (XceliumML)
  - 6x - 17x speed-up factor
  - $\approx 99.9\%$ coverage regain

**Radar based SoC**
- System Verilog design
- UVM-SV testbench
- Ongoing project
- 25 tests/ 271 runs @ Stage I
- 35 tests/ 301 runs @ Stage II

- Results (XceliumML):
  - 3x speed-up factor
  - Over 100% coverage regain
  - Able to discover more functional bugs
Agenda

• Motivation and Problem Statement
• Cadence Xcelium ML
• Application on Designs
• Xcelium ML vs Ranking
• Proposed Methodology
• Summary
Xcelium ML vs Ranking

**Xcelium ML**
- ML tries to find the correlations between the actions inside the testbench and the coverage results
- The produced regression by ML is randomized in nature
- It can improve/ increase the original coverage
- It can discover new functional bugs that the orig regr did not find

**Ranking**
- Ranking finds the effective test and seed pairs to reach the same coverage
- Re-simulation of tests → Directed testing
- Inability to exercise new scenarios
  - No new coverage
  - Can not identify the bugs
Xcelium ML vs Ranking

Coverage Regain
Xcelium ML vs Ranking

Various Projects
- Microprocessor IP (Stage I)
- Microprocessor IP (Stage II)
- Mixed Signal SoC
- Radar based SoC (Stage I)
- Radar based SoC (Stage II)

Coverage Regain in %
- Microprocessor IP (Stage I): 100%
- Microprocessor IP (Stage II): 100%
- Mixed Signal SoC: 99.89%
- Radar based SoC (Stage I): 100%
- Radar based SoC (Stage II): 108.42%

Comparison:
- Ranking
- Xcelium ML
Xcelium ML vs Ranking

Compression Factor in Runtime
Xcelium ML vs Ranking

Compression Factor in Runs
Xcelium ML vs Ranking
Agenda

• Motivation and Problem Statement
• Cadence Xcelium ML
• Application on Designs
• Xcelium ML vs Ranking
• Proposed Methodology
• Summary
Proposed Methodology using Xcelium ML
Agenda

• Motivation and Problem Statement
• Cadence Xcelium ML
• Application on Designs
• Xcelium ML vs Ranking
• Proposed Methodology
• Summary
Summary

- Out of the box solution with minimal setup
- Time & cost savings during simulation regression
- 3x to 15x speed-up factor
- Generates a compressed regression with random seeds (unlike Ranking)
  - Uncover new functional bugs
  - Might exceed original coverage
- ML learning/ training consumes extra 20% (on average) of the original regression time to produce an optimized regression
- Re-learning required after significant RTL/ TB changes (typically every week)
Thank you.