Machine Learning Driven Verification
A Step Function in Productivity and Throughput

Daniel Hansson
John Rose
Matt Graham
Agenda

• What is Machine Learning?
• Machine Learning for Formal, Simulation, and Regression Testing
• Improved Bug Hunting Efficiency with Machine Learning
• Leveraging Machine Learning for Automatic Debug of Regression Failures
• Summary and Wrap Up
• Q&A
What is Machine Learning?

How can it shorten the verification cycle?
Intelligence demonstrated by machines, as opposed to natural intelligence, displayed by animals including humans
Algorithm

- A finite sequence of well-defined instructions, typically used to solve a class of specific problems or to perform a computation
Machine learning (ML) is the study of computer algorithms that can improve automatically through experience and by use of data. It is seen as a part of artificial intelligence.

Artificial Neural Networks, Decision Trees, Support-Vector Machines, Genetic Algorithms
Machine Learning or Automation?

vManager™ Verification Management Test Weight Optimization

- Test 1 with 25 random seeds
- Test 2 with 25 random seeds
- Test 3 with 25 random seeds

iteration

75 runs
10.3%

75 runs
22.5%

75 runs
57.8%

Higher coverage and effectiveness

Automated ✓
Algorithmic ✓
Learning ✗
Machine Learning for Formal, Simulation, and Regression
Automation = Throughput

ROI Mindset: Bug closure per $ per day

VERIFICATION THROUGHPUT
ML Application in Verification

- Bug Prediction
- Failure Triage
- Bug Hunting
- Coverage Regain
- Optimal Engine Selection

© 2022 Cadence Design Systems, Inc. All rights reserved.
Improved Bug Hunting Efficiency
Xcelium Machine Learning for Verification Efficiency

• Cadence® Xcelium™ ML
  • Trains on large set of regression runs
  • Creates runs to more efficiently hit coverage

Learning models

Input Layer: random control variables
Hidden Layers
Output Layer: coverage bin

Coverage Focus

Random Controls

Learning models

© 2022 Cadence Design Systems, Inc. All rights reserved.
What can you do with Xcelium Machine Learning?

**Regression Compression**

<table>
<thead>
<tr>
<th>Original Regression</th>
<th>ML Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bins Covered</strong></td>
<td><strong>Bins Covered</strong></td>
</tr>
<tr>
<td>393226</td>
<td>390528</td>
</tr>
<tr>
<td><strong>CPU Time</strong></td>
<td><strong>CPU Time</strong></td>
</tr>
<tr>
<td>10052 cpuH</td>
<td>1950 cpuH</td>
</tr>
</tbody>
</table>

**Targeted Regression**

Original regression for full design

Augmenting runs from ML

**Bug Hunting / Coverage Closure**

Original Regression

Augmenting runs from ML

prop_1 25%
prop_2 60%
prop_3 5%
prop_4 25%

prerequisite cover properties

prop_1 35%
prop_2 60%
prop_3 15%
prop_4 40%

390528 1950 cpuH 99.3% 5.1x

Original Regression

CPU Time

393226 10052 cpuH

Regression Compression

© 2022 Cadence Design Systems, Inc. All rights reserved.
Using Machine Learning for Bug Hunting

- Typical bug-hunting using randomized testbenches
- Once bug rate reaches some low threshold
- Fill CPU resources with random runs
- Increase seeds for tests most likely to hit unique scenarios

Random config / Scenario gen

Front-end state

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>C</td>
<td>C</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
<td>A</td>
<td>D</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

... N: B | B | C | A | B | C | B | A | C | ... |

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>A</td>
<td>C</td>
<td>D</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>B</td>
<td>D</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>D</td>
<td>B</td>
<td>C</td>
<td>B</td>
</tr>
</tbody>
</table>

Random sequences

Bug signature

Rare scenarios come close to bug exposure but not quite

© 2022 Cadence Design Systems, Inc. All rights reserved.
Using Machine Learning for Bug Hunting

- Machine Learning bug hunting using randomized testbenches
  - Focus on front-end states that magnify more rare conditions
Using Machine Learning for Bug Hunting

- Augment full regression with ML-generated runs
  - The ML-generated regression will create higher percentage of more rare scenarios
  - The bug rate of the ML runs (unique signature / cpuH) will typically be higher than the full regression
  - Use in conjunction with the full regression until the full regression no longer finds new bug signatures

![Project timeline](image)

- Original Regression
- ML augmentation
- Cumulative rate
Leveraging Machine Learning for Automatic Debug of Regression Failures
**PinDown-ML**  
**Regression Flow Improvement**

**Traditional Process**
- Team updates design: Early evening
- Regression Runs: Overnight
- Triage test failures: Large part of the morning
- Bad commit found: Midday sometime

**Process with PinDown-ML**
- Team updates design: Early evening
- PinDown-ML bug prediction
- PinDown-ML automated regression debug
- Bad commit automatically found: Early in Morning

**Saves time and effort!**

Savings measured in a customer ASIC project (40 engineers, 1 year)
- 4.9 work years
- Bugs fixed 4X faster => 11% shorter project time
PinDown-ML
Automatic Debugger of Regression Failures

PinDown-ML automates debugging of regression failures and sends out bug reports such as this.

1. Fast Debug
   Uses machine learning to predict bugs before simulation starts.

2. Quality Reports
   Validates bug report by repairing faulty code to make the test pass again before bug report is issued.

3. High Granularity
   Can show the exact line of code that is faulty.

Example Bug report from PinDown-ML:
Bug Prediction

Revision History
Code
Logs
Test Results
Bug Reports

Machine Learning
Training

TEST PASS

Bug Prediction Model

TEST FAIL

Repair Code to Validate Bug

Issue Bug Report

© 2022 Cadence Design Systems, Inc. All rights reserved.
PinDown-ML
Feature: Commit Time

Median insertion time for bugs: 3 pm
PinDown-ML
Feature: File Ownership

LOW RISK
Single Committer

Many Committers

HIGH RISK
Everyone
Result: 41% Precision

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision Mean</td>
<td>0.415</td>
</tr>
<tr>
<td>Precision Standard Deviation</td>
<td>0.0456</td>
</tr>
<tr>
<td>Recall Mean</td>
<td>0.186</td>
</tr>
<tr>
<td>Recall Standard Deviation</td>
<td>0.0255</td>
</tr>
</tbody>
</table>

=> 96% chance bug is in top 6 commits

Source: Poster/Paper “Predicting Bad Commits” from DVCon US, Feb 2019
PinDown-ML
Measured at Customer: 51% of bugs validated at first attempt owing to good bug prediction

This shows that our ML-based bug prediction works very well in real life.

51% of validated bugs have a ranking of 1-3
This means that 51% of bugs are validated in one single iteration (because 3 slots are reserved for validation).
**PinDown-ML**

**NEW! No extra setup required for vManager users**

If you run regressions through vManager today, no extra setup is necessary for PinDown to rerun tests during debug.

PinDown communicates directly with vManager through vAPI.
Machine Learning for Verification at Cadence

Jasper, Xcelium, and vManager
Jasper ProofMaster Smart Proof Automation
Automates expert-level optimizations

- **Proof Profiling Data**
  - Keep engine-level settings that worked before

- **Proof Caching**
  - Reuse existing result if constraints and COI unchanged

- **Multi-Advisor Proof Orchestration**
  - Use Machine Learning to find best macro-level settings

---

- Optimizes subsequent runs/regressions
  - Find more bugs
  - Better convergence
  - Faster proofs

---

- Optimizes out-of-the-box proofs

---

“What the EDA users REALLY think

"The tool uses machine learning... ...and the results are on par with careful selection by a human"
Machine Learning in the Cadence Verification Flow

- PinDown-ML
- Bug Prediction
- Failure Triage
- Coverage Regain
- Bug Hunting
- Proof Orchestration

Xcelium™

Jasper™ Formal

© 2022 Cadence Design Systems, Inc. All rights reserved.
Summary and Wrap Up
Summary

• Improved verification throughput via automation is a necessity.

• Opportunity to apply Machine Learning
  • Not all automation is ML

• Significant potential for ML in Bug Hunting and Bug Prediction

• Cadence is applying ML across the verification solution
Q&A
Thank You!

• Additional Questions?

• John Rose – jlrose@cadence.com
• Daniel Hansson – hansson@cadence.com
• Matt Graham – magraham@cadence.com