2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Machine Learning Driven Verification
A Step Function in Productivity and Throughput

Daniel Hansson

John Rose

Matt Graham

cadence

accellera SYSTEMS INITIATIVE

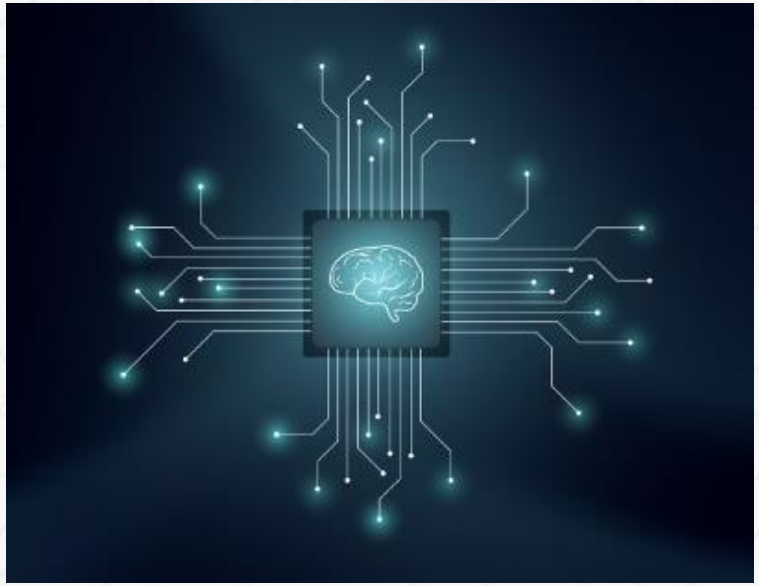
Agenda

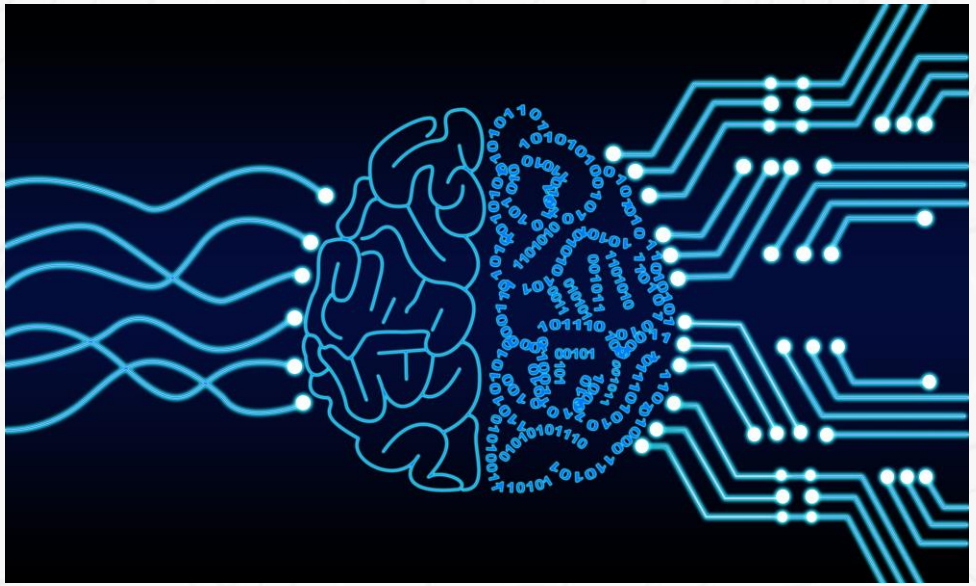
- What is Machine Learning?
- Machine Learning for Formal, Simulation, and Regression Testing
- Improved Bug Hunting Efficiency with Machine Learning
- Leveraging Machine Learning for Automatic Debug of Regression Failures
- Summary and Wrap Up
- Q&A

What is Machine Learning?

How can it shorten the verification cycle?

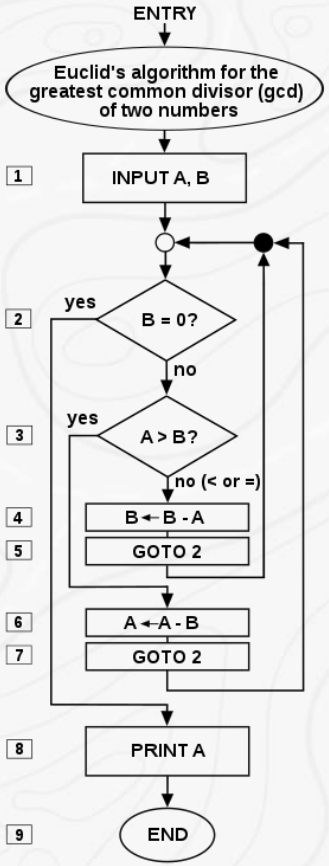
Artificial Intelligence

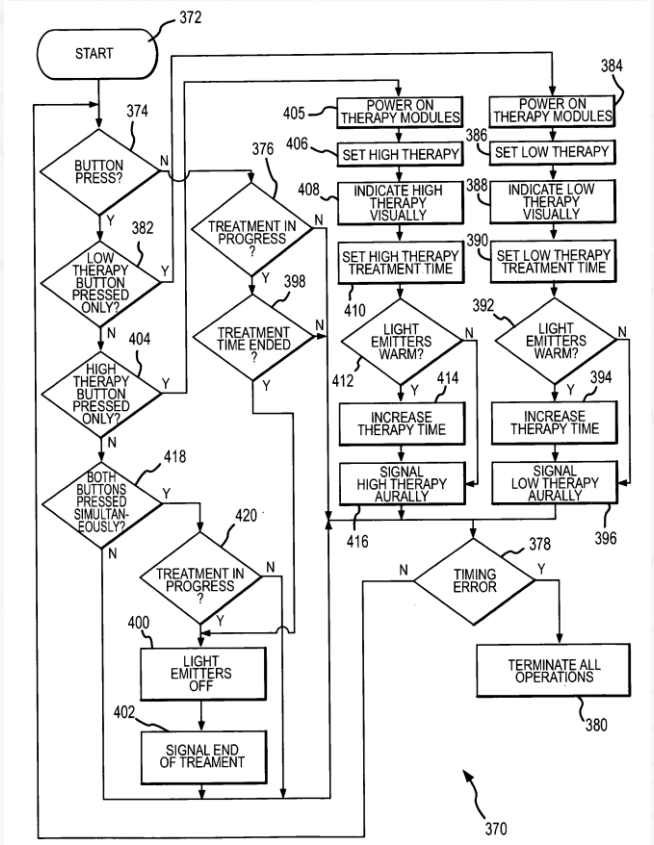




Intelligence demonstrated by machines, as opposed to natural intelligence, displayed by animals including humans

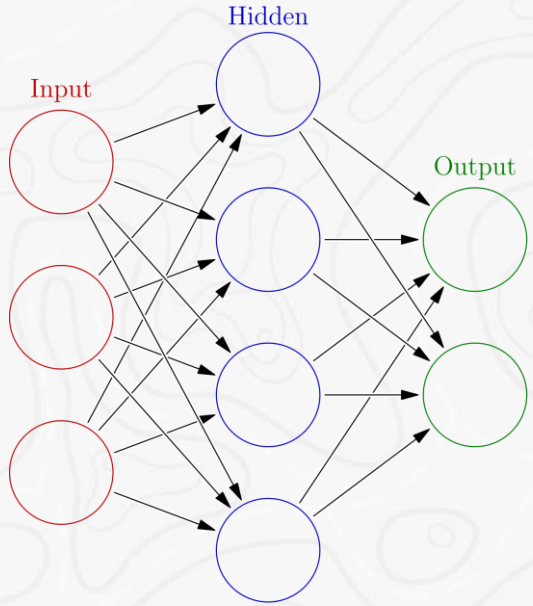
Algorithm

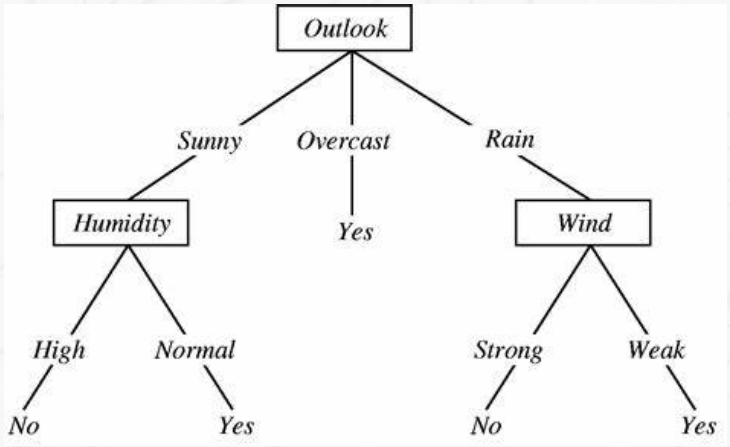


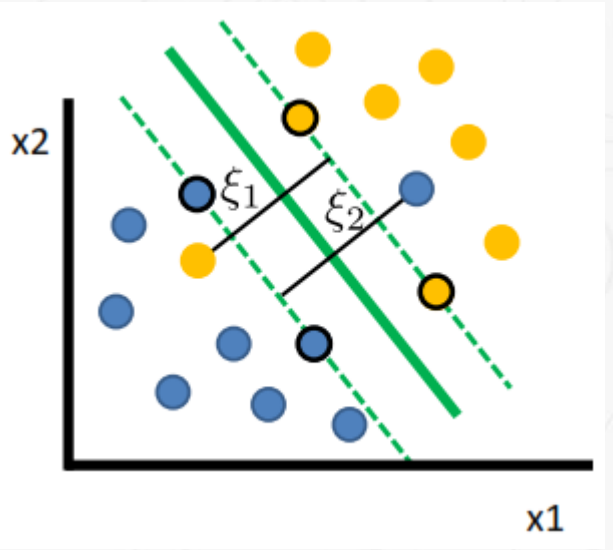


- A finite sequence of well-defined instructions, typically used to solve a class of specific problems or to perform a computation

Machine Learning





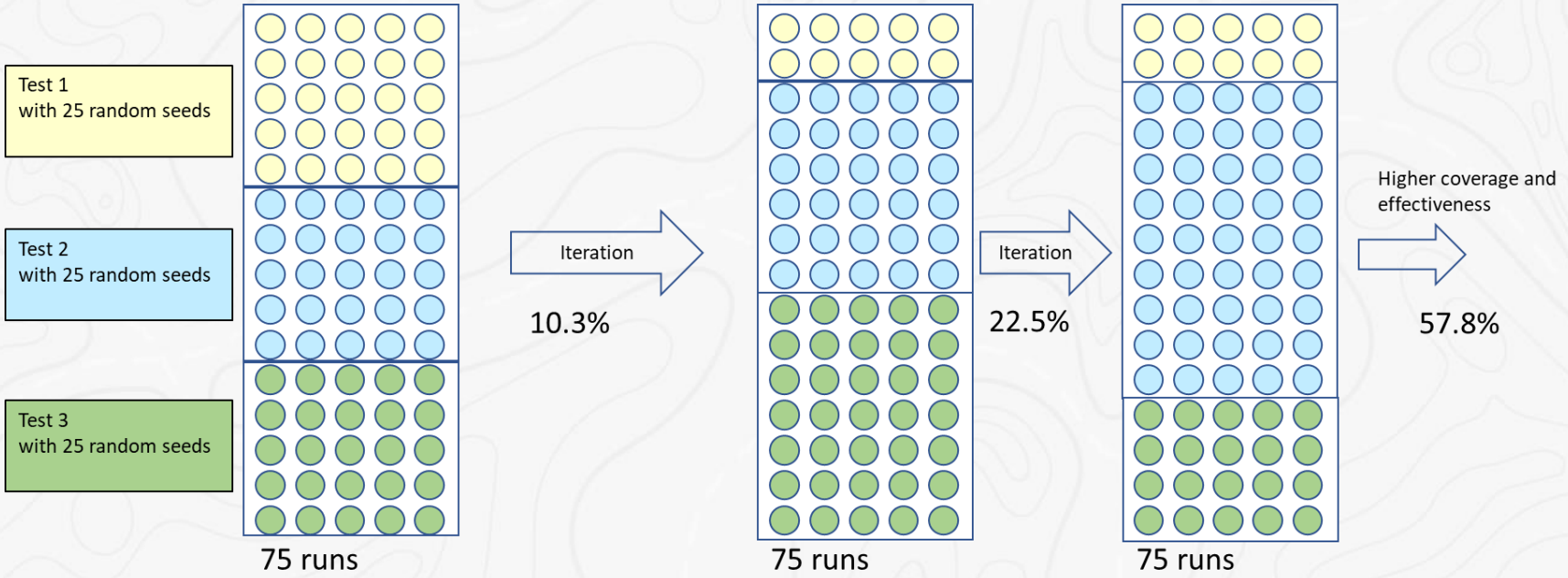


Machine learning (ML) is the study of **computer algorithms** that can **improve automatically** through **experience** and by **use of data**. It is seen as a part of artificial intelligence.

Artificial Neural Networks, Decision Trees, Support-Vector Machines, Genetic Algorithms

Machine Learning or Automation?

vManager™ Verification Management Test Weight Optimization



Automated ✓

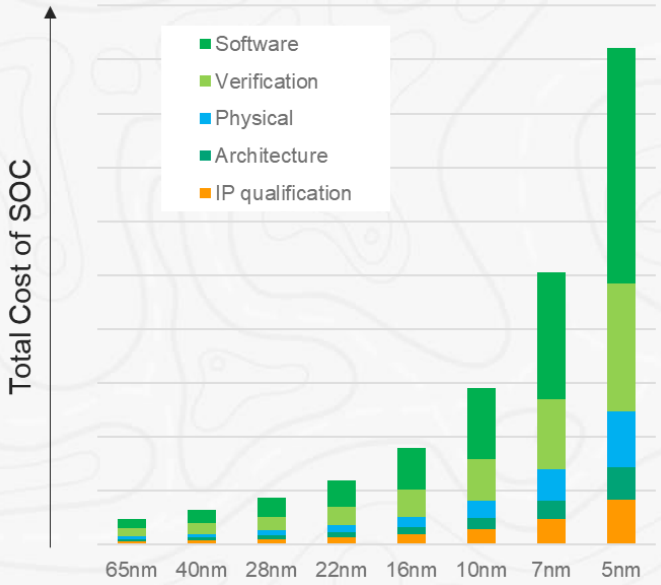
Algorithmic ✓

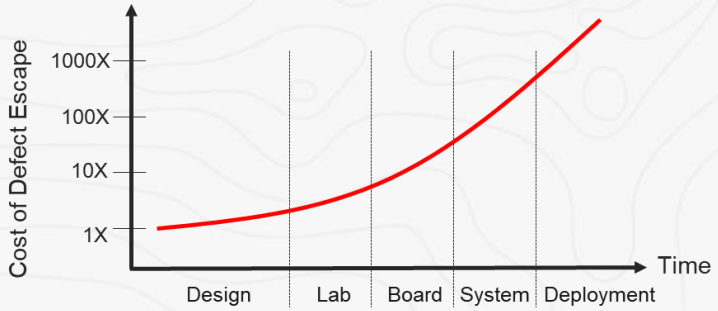
Learning ✗

Machine Learning for Formal, Simulation, and Regression

Automation = Throughput

Exponential Challenge





ROI Mindset: Bug closure per $ per day

VERIFICATION THROUGHPUT

ML Application in Verification

Bug Prediction

Coverage Regain

Failure Triage

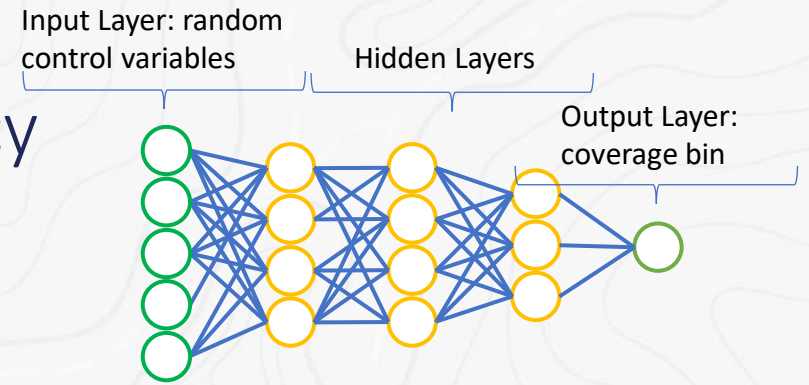
Optimal Engine Selection

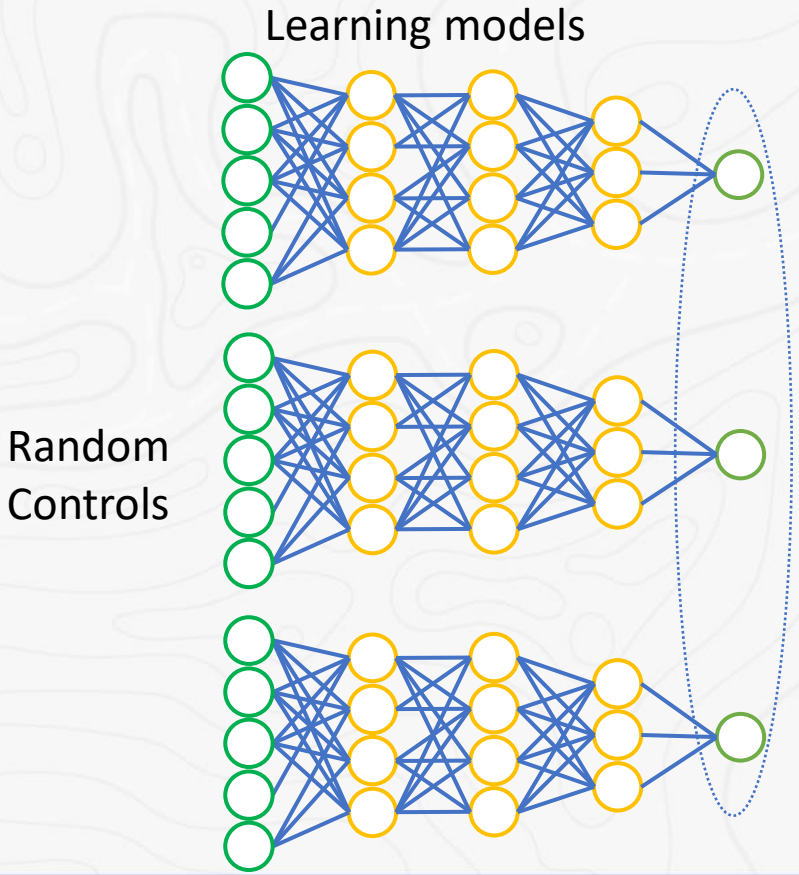
Bug Hunting

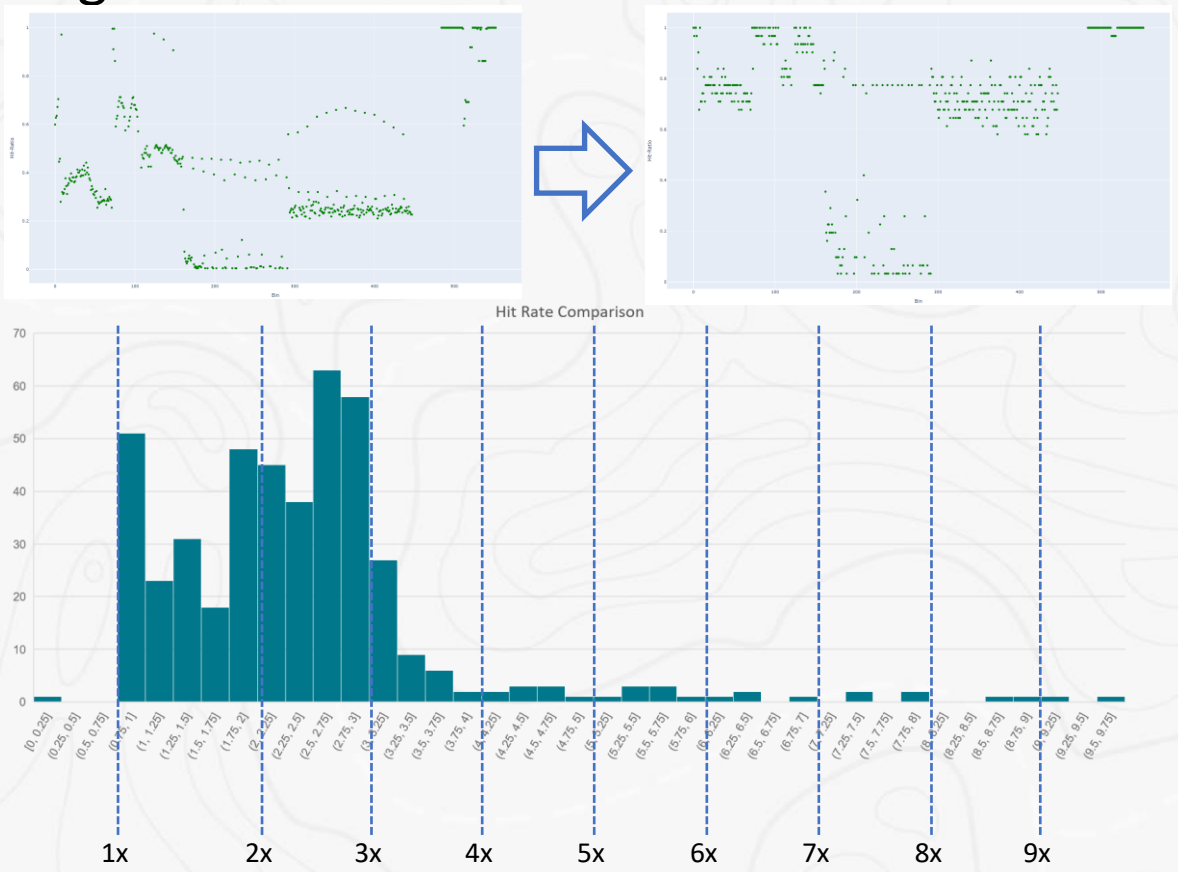
Improved Bug Hunting Efficiency

Xcelium Machine Learning for Verification Efficiency

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







What can you do with Xcelium Machine Learning?

Regression Compression

Original Regression

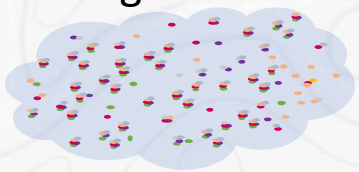
Bins Covered	CPU Time
393226	10052 cpuH

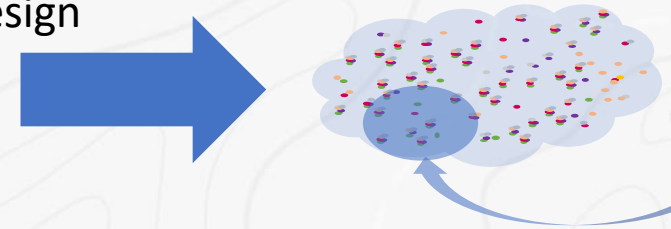
ML Regression

Bins Covered	CPU Time	Regain	Compression
390528	1950 cpuH	99.3%	5.1x

Targeted Regression

Original regression for full design

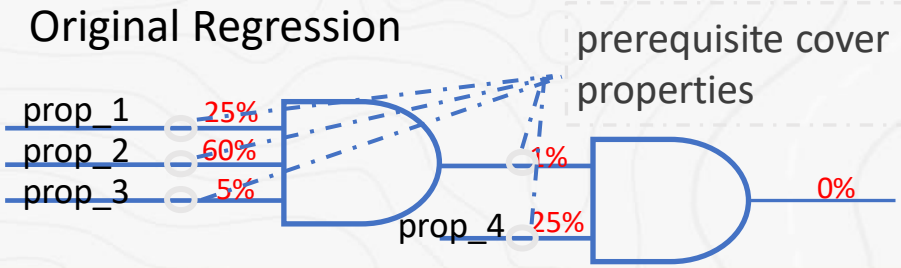




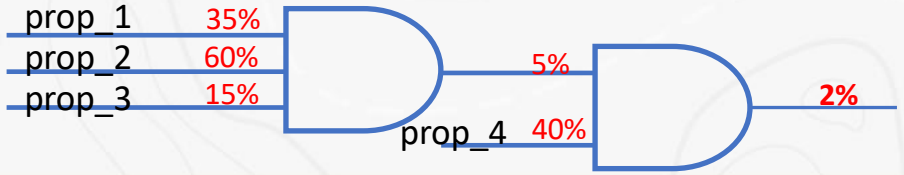
Augmenting runs from ML

Bug Hunting / Coverage Closure

Original Regression

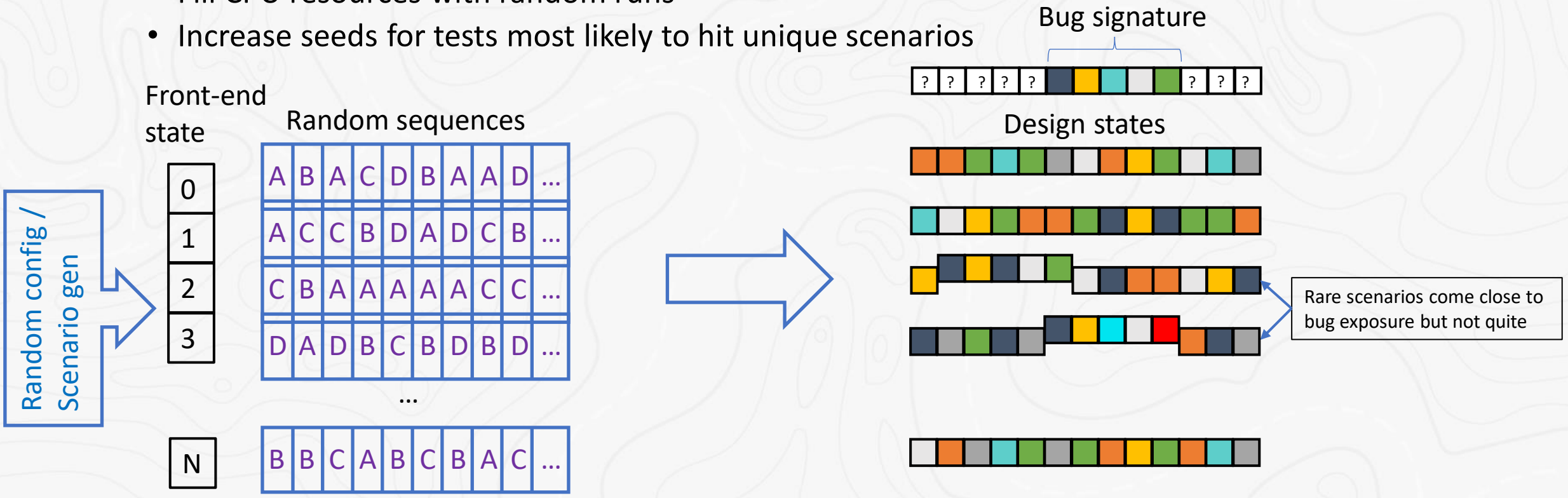


ML Regression focused runs



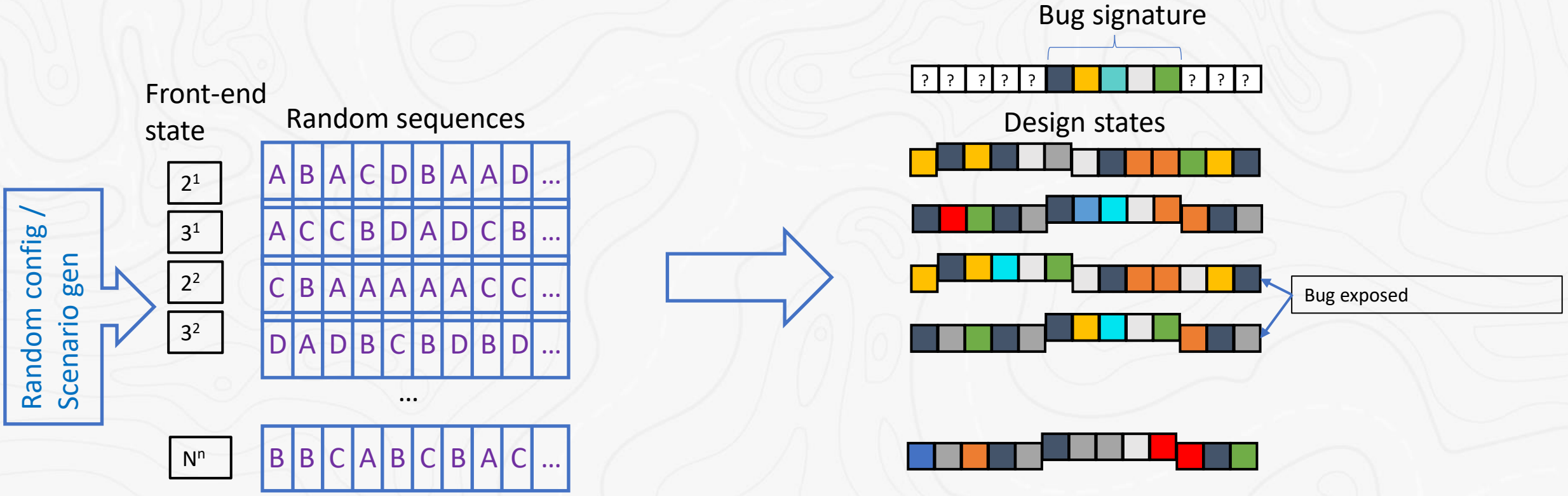
Using Machine Learning for Bug Hunting

- 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



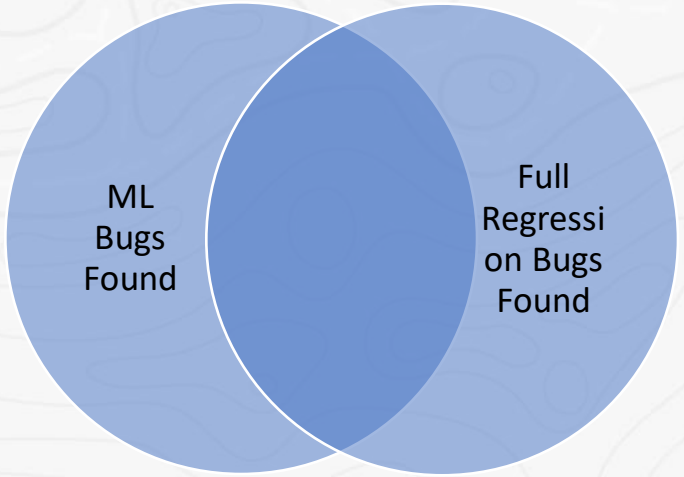
Using Machine Learning for Bug Hunting

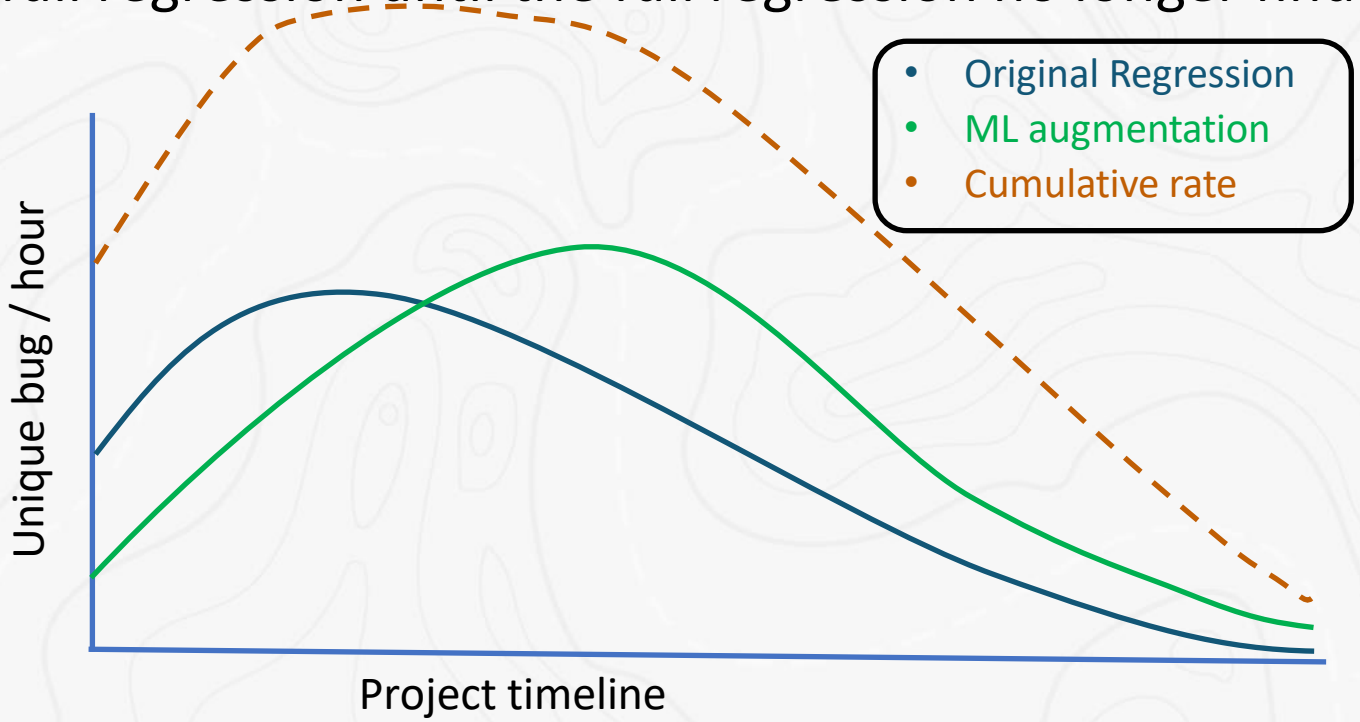
- Machine Learning bug hunting using randomized testbenches
  - Focus on front-end states that magnify more rare conditions



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

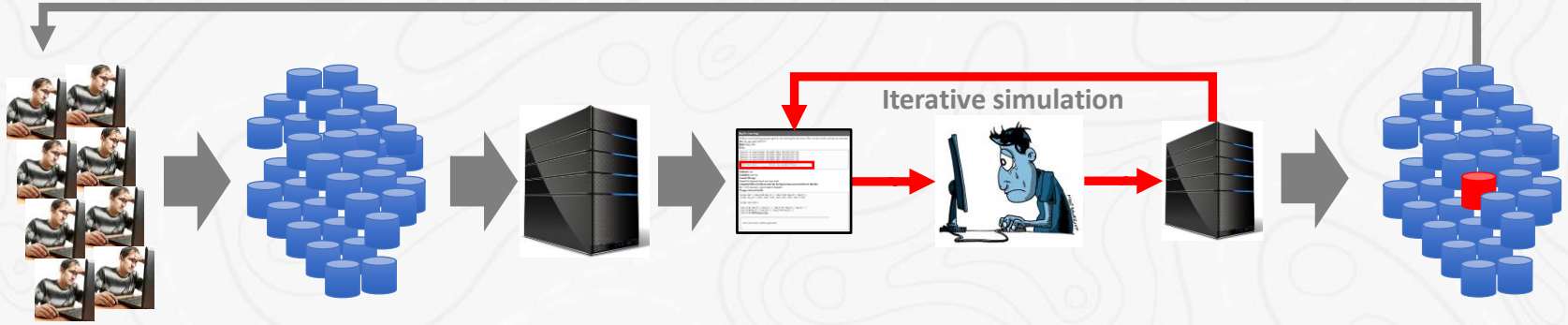




Leveraging Machine Learning for Automatic Debug of Regression Failures

PinDown-ML
Regression Flow Improvement

Traditional Process



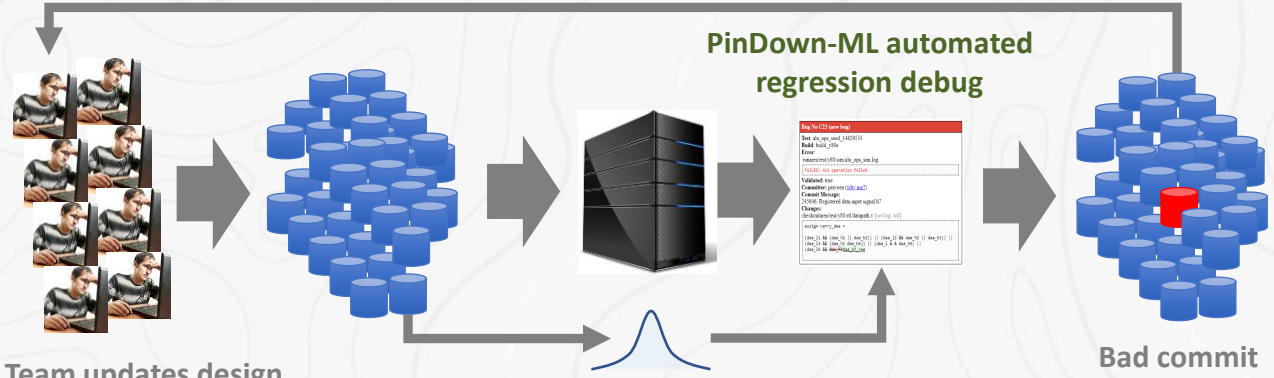
Team updates design
Early evening

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

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

Bug No C23 (new bug)

Test: alu_ops_seed_14829533
Build: build_y80e
Error:
runarea/test/y80/sim/alu_ops_sim.log

```
FAILED: ALU operation failed
```

Validated: true
Committer: praveen (why me?)
Commit Message:
245646. Registered data input signal h7
Changes:
checkoutarea/test/y80/rtl/datapath.v [*verilog, hdl*]

```
assign carry_daa =

(daa_l1 && (daa_h1 || daa_h2)) || (daa_l2 && daa_h2 || daa_h3)) ||
(daa_l3 && (daa_h1 daa_h4)) || (daa_l & & daa_h5( ||
(daa_l5 && daa_h7daa_h7_reg
```

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

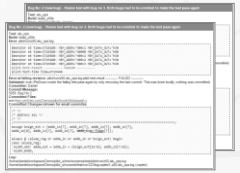
Bug Prediction

Revision History

Code

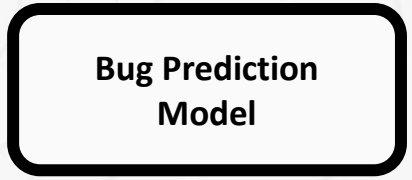
Logs

Test Results



Bug Reports

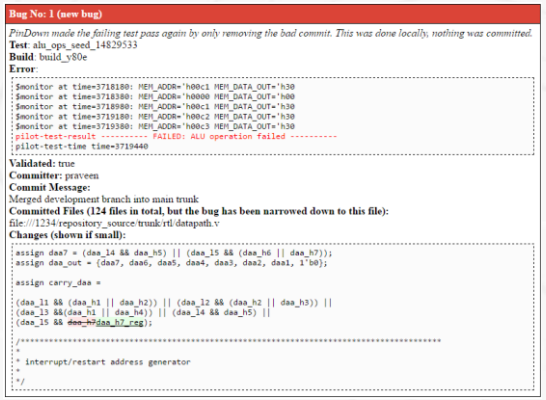
TEST FAIL



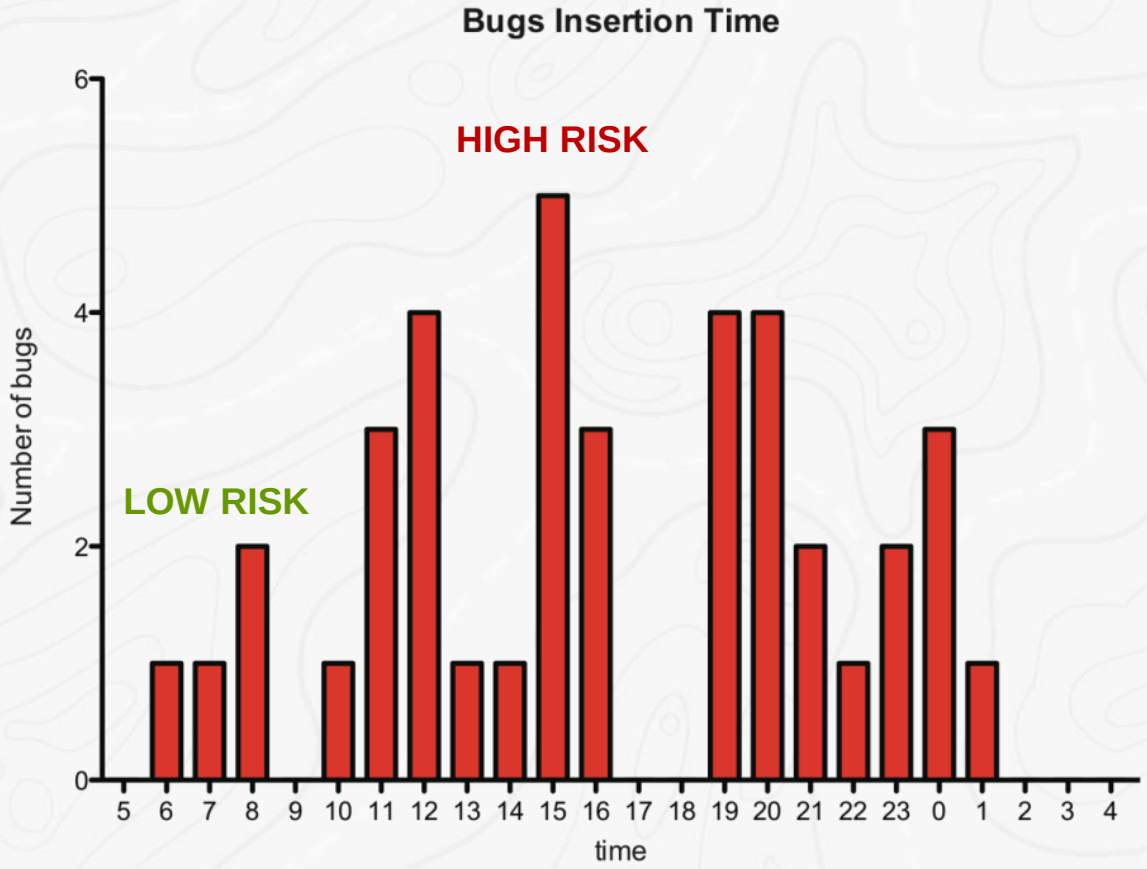
Repair Code to Validate Bug

TEST PASS

Issue Bug Report

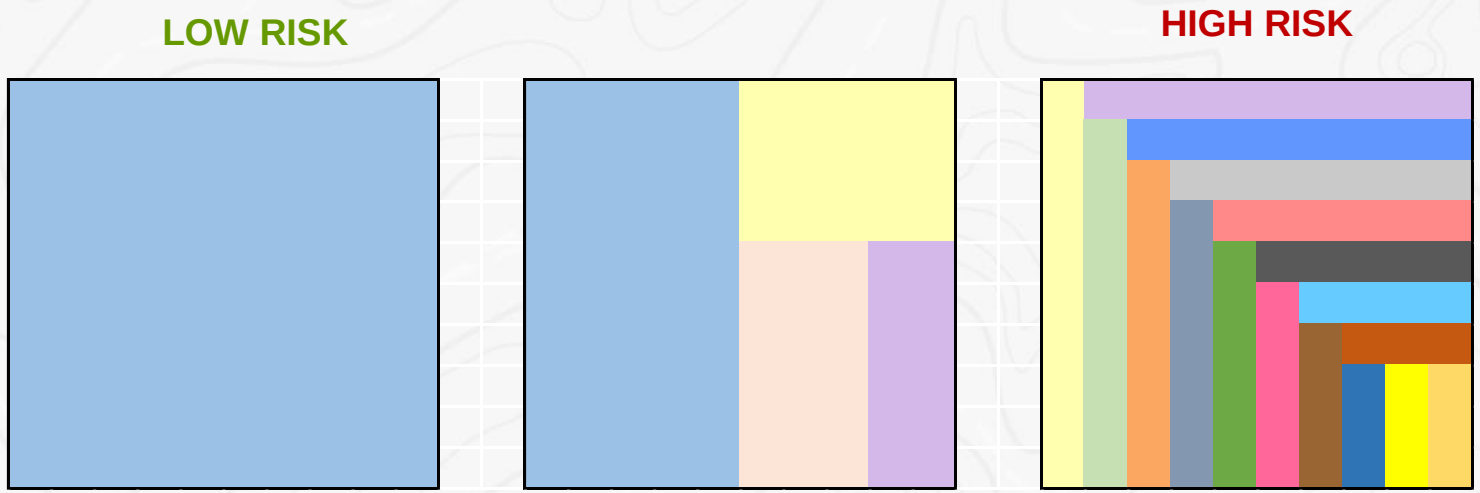


PinDown-ML
Feature: Commit Time



Median insertion time for bugs: 3 pm

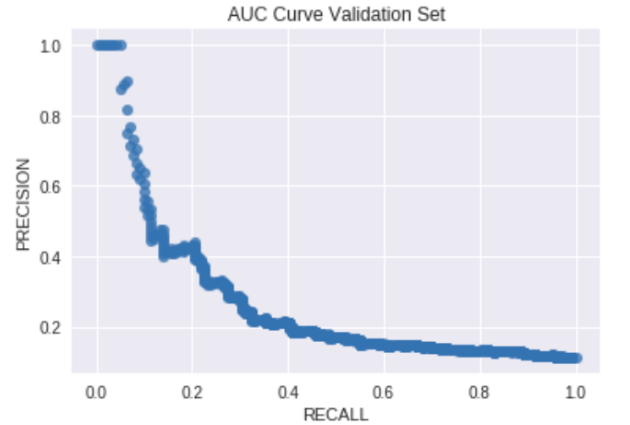
PinDown-ML
Feature: File Ownership



Result: 41 % Precision

41% Precision

Metric	Value
Precision Mean	0.415
Precision Standard Deviation	0.0456
Recall Mean	0.186
Recall Standard Deviation	0.0255



=> 96% chance bug is in top 6 commits

Bug Predictions

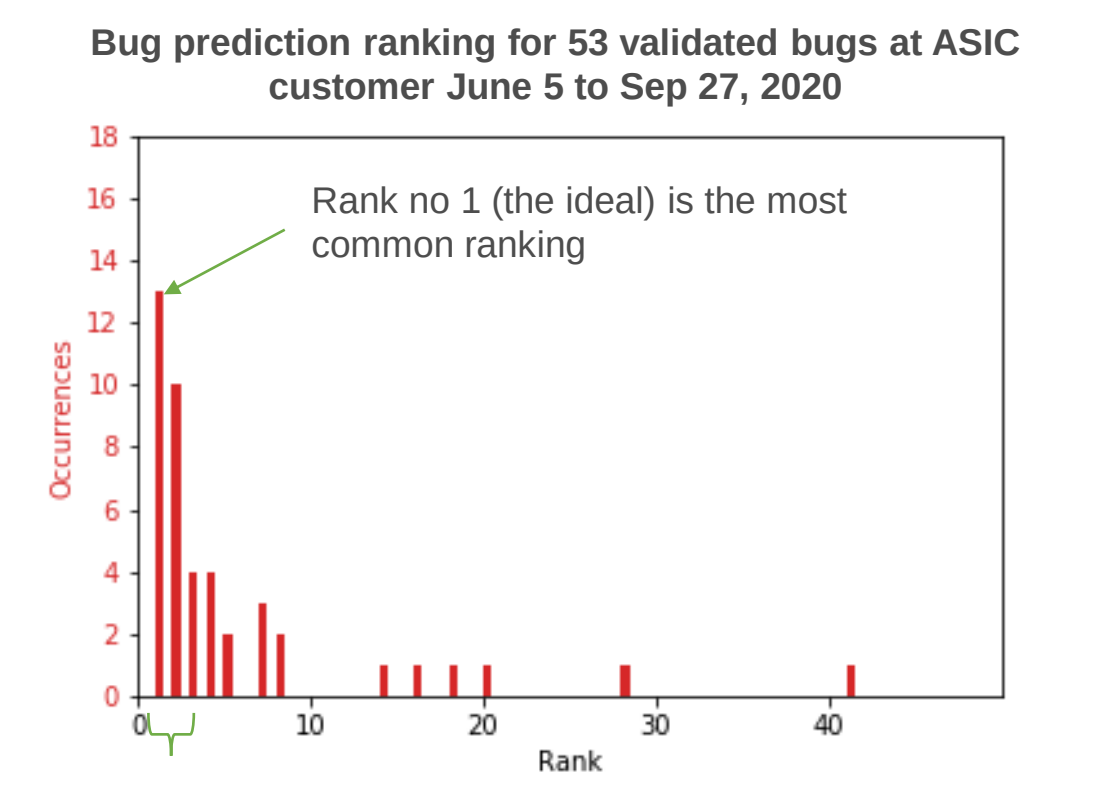
Prediction	Revision	Date	Committer	Commit message
0.999403	...stbed0.git:7c643333a5	Feb 12 2011 5:19 AM CET	carlos	repo1@7 change c2t3: c2t6: c2t5:
0.996919	...stbed0.git:f36f5c8981	Feb 12 2011 5:16 AM CET	prashant	repo1@6 change c2t3: result (bit 0) to F c2t6: result (bit 0) to F c2t5: result (bit 0) to F
0.759582	...stbed0.git:26c2a86713	Feb 12 2011 5:25 AM CET	nageshwar	repo1@9 change config_2: result (bit 0) to P
0.759582	...stbed0.git:18e57d6808	Feb 12 2011 5:13 AM CET	sharon	repo1@5 change config_2: result (bit 0) to F
0.620521	...stbed1.git:816246c63f	Feb 12 2011 5:17 AM CET	praveen	repo2@6 change c2t3: c2t6: c2t5:
0.587064	...stbed2.git:54abe25cc2	Feb 12 2011 5:21 AM CET	prashant	repo3@7 change c2t2: result (bit 0) to F c2t1: result (bit 0) to F c2t4: result (bit 0) to F
0.043611	...stbed2.git:5800e5fee3	Feb 12 2011 5:18 AM CET	carlos	repo3@6 change c2t3: c2t6: c2t5:
0.000000	...stbed1.git:f0679c2296	Feb 12 2011 5:14 AM CET	hemal	repo2@5 change empty update
0.000000	...stbed1.git:c48c888f86	Feb 12 2011 5:05 AM CET	nageshwar	repo2@2 change empty update
0.000000	...stbed1.git:463089ee6b	Feb 12 2011 5:26 AM CET	prashant	repo2@9 change empty update
0.000000	...stbed1.git:a8f0c9ff4d	Feb 12 2011 5:11 AM CET	prashant	repo2@4 change empty update
0.000000	...stbed2.git:2255f0208e	Feb 12 2011 5:06 AM CET	prashant	repo3@2 change empty update
0.000000	...stbed2.git:5da6eb734b	Feb 12 2011 5:33 AM CET	sharon	repo3@11 change empty update
0.000000	...stbed2.git:16e699db5d	Feb 12 2011 5:30 AM CET	carlos	repo3@10 change empty update
0.000000	...stbed2.git:2286b531e8	Feb 12 2011 5:12 AM CET	sharon	repo3@4 change empty update
0.000000	...stbed2.git:a936a95dbc	Feb 12 2011 5:09 AM CET	carlos	repo3@3 change empty update
0.000000	...stbed1.git:1e01c3e2d0	Feb 12 2011 5:32 AM CET	sharon	repo2@11 change empty update
0.000000	...stbed1.git:f3816f9638	Feb 12 2011 5:29 AM CET	carlos	repo2@10 change empty update
0.000000	...stbed1.git:e58f2c0b20	Feb 12 2011 5:23 AM CET	sharon	repo2@8 change empty update
0.000000	...stbed1.git:9edc996e9e	Feb 12 2011 5:20 AM CET	carlos	repo2@7 change empty update
0.000000	...stbed1.git:7559f5647d	Feb 12 2011 5:08 AM CET	carlos	repo2@3 change empty update
0.000000	...stbed2.git:70cd69b001	Feb 12 2011 5:27 AM CET	praveen	repo3@9 change empty update
0.000000	...stbed2.git:6ff1c0be08	Feb 12 2011 5:24 AM CET	nageshwar	repo3@8 change empty update
0.000000	...stbed2.git:c9537c0d63	Feb 12 2011 5:15 AM CET	hemal	repo3@5 change empty update
0.000000	...stbed0.git:e46bf2274d	Feb 12 2011 5:07 AM CET	praveen	repo1@3 change empty update
0.000000	...stbed0.git:97aeededc1	Feb 12 2011 5:22 AM CET	sharon	repo1@8 change empty update
0.000000	...stbed0.git:28f0f5ed39	Feb 12 2011 5:04 AM CET	nageshwar	repo1@2 change empty update
0.000000	...stbed0.git:9d0fb28415	Feb 12 2011 5:31 AM CET	prashant	repo1@11 change empty update
0.000000	...stbed0.git:55bf132a5c	Feb 12 2011 5:28 AM CET	carlos	repo1@10 change empty update
0.000000	...stbed0.git:3a5b0dc8fc	Feb 12 2011 5:10 AM CET	carlos	repo1@4 change empty update

Generated by PinDown v4.2.5acf331+ Jul 9 2018 2:46 PM CEST

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



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)

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

PinDown-ML

NEW! No extra setup required for vManager users

vManager™
Total Verification Management

rerun

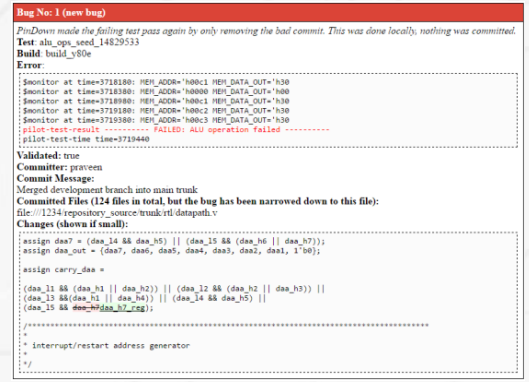
vAPI

results

PinDown-ML

report

Bug Report



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

ProofMaster
Component and Data Management

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

What the EDA users REALLY think
DEEPCHIP

"The tool uses machine learning...
…and the results are on par with careful selection by a human"

Learning

Machine Learning

Optimizes subsequent runs/regressions

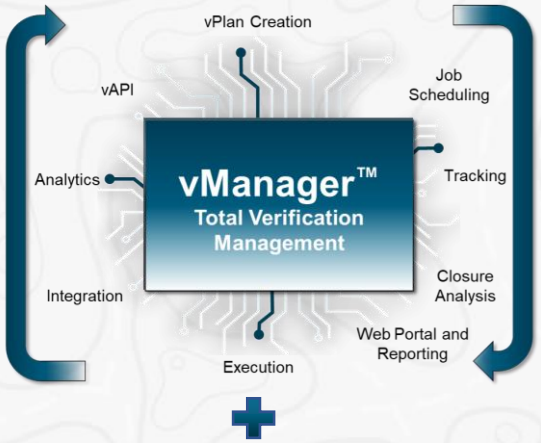
Optimizes out-of-the-box proofs

Find more bugs

Better convergence

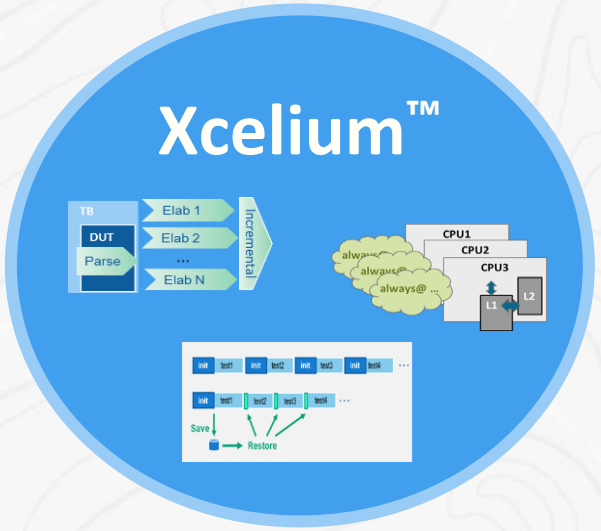
Faster proofs

Machine Learning in the Cadence Verification Flow

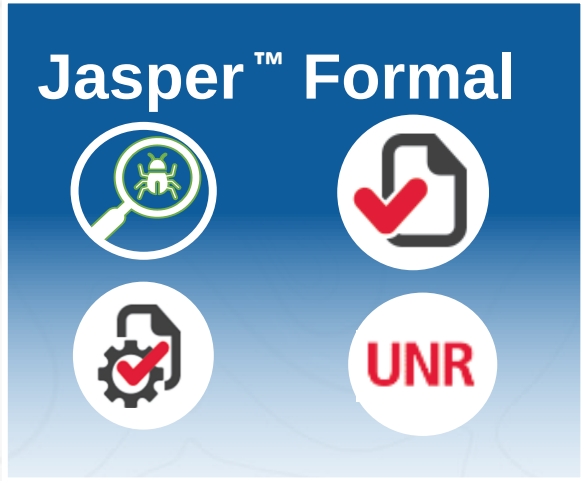


PinDown-ML

Bug Prediction
Failure Triage



Coverage Regain
Bug Hunting



Proof
Orchestration

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