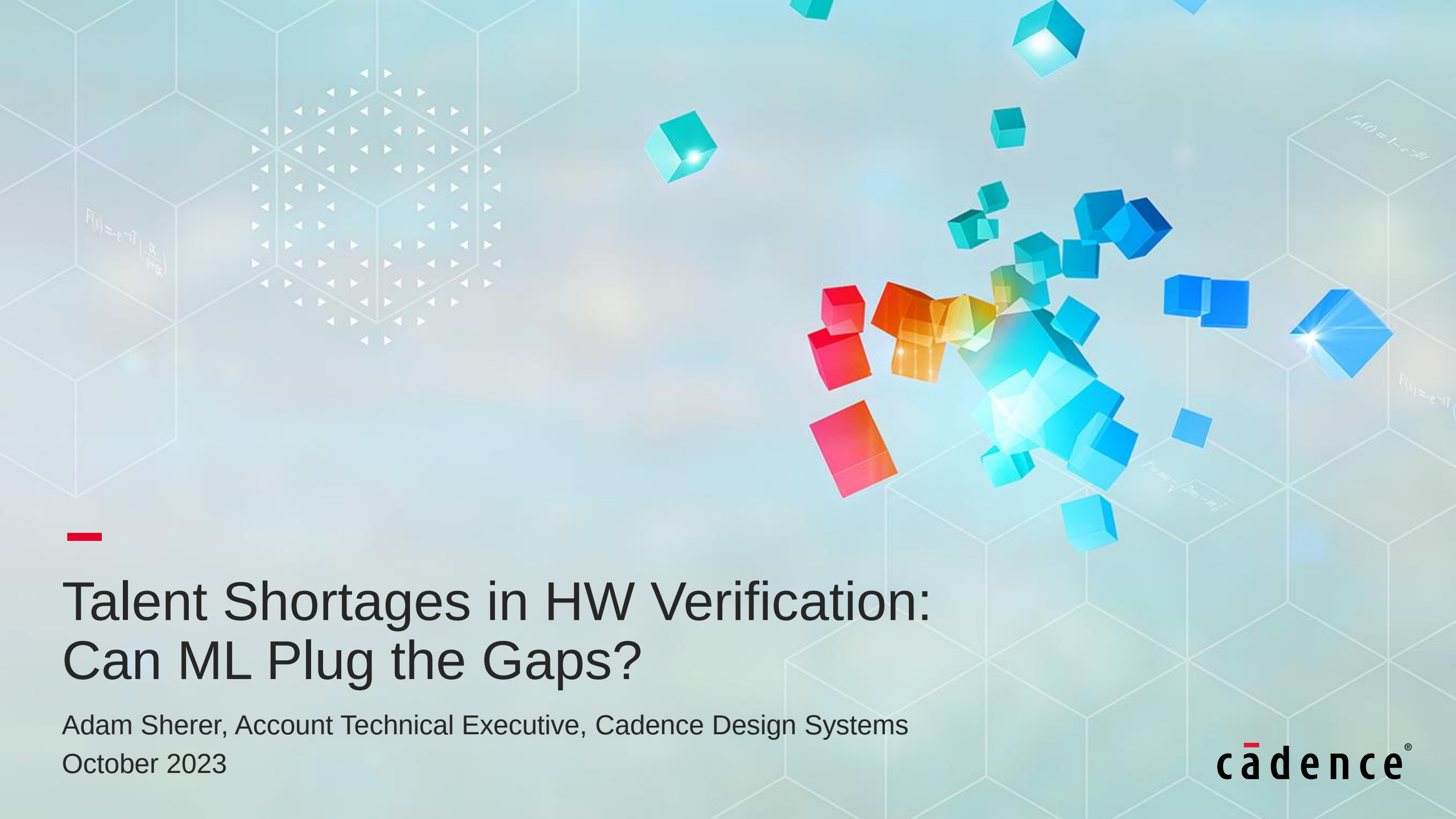


Talent Shortages in HW Verification: Can ML Plug the Gaps?

Adam Sherer, Account Technical Executive, Cadence Design Systems

October 2023

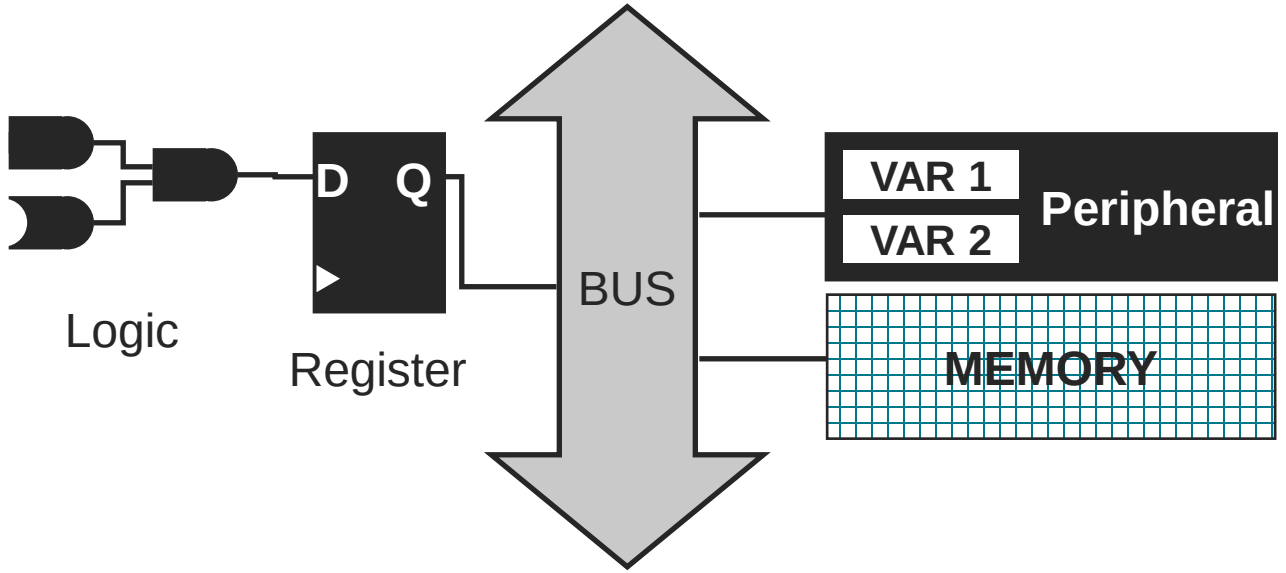
cadence®

Agenda

- Verification complexity outpacing engineering availability
- Deepening data deluge defines debug doldrums
- Machine learning opportunities to enable talent efficiency

Verification Complexity Grows Exponentially

And available verification engineers grows arithmetically



- Verification scales exponentially with states of design: 2^(storage elements)
  - Includes latches, registers, flops, memory-mapped I/O, and embedded SW data memory
- Complete verification: only enter specified states and never enter any other state
- Pragmatic goal is comprehensive verification
  - 100K gate pure digital design may contain 10K storage bits = 2^10000 possible states
  - Multi-FPGA / ASIC systems are 10's of millions to billions of storage bits

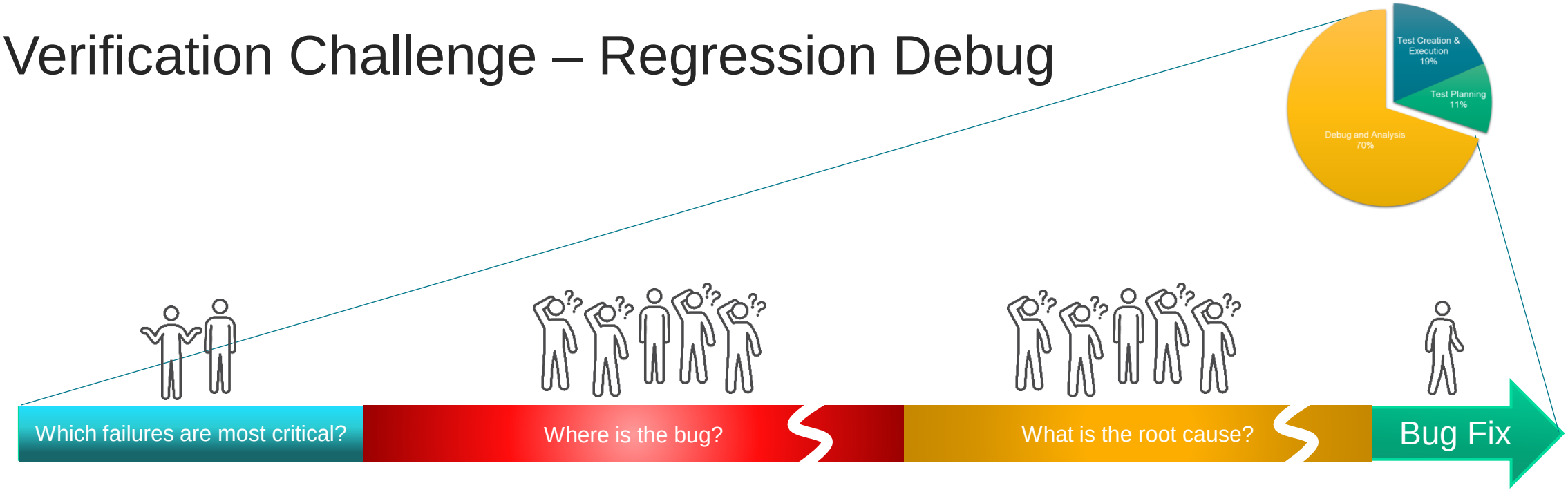
Engineering Efficiency Lost Throughout the Flow

Efficiency analyzing failures, root cause identification

Efficiency executing up to millions of tests in regression

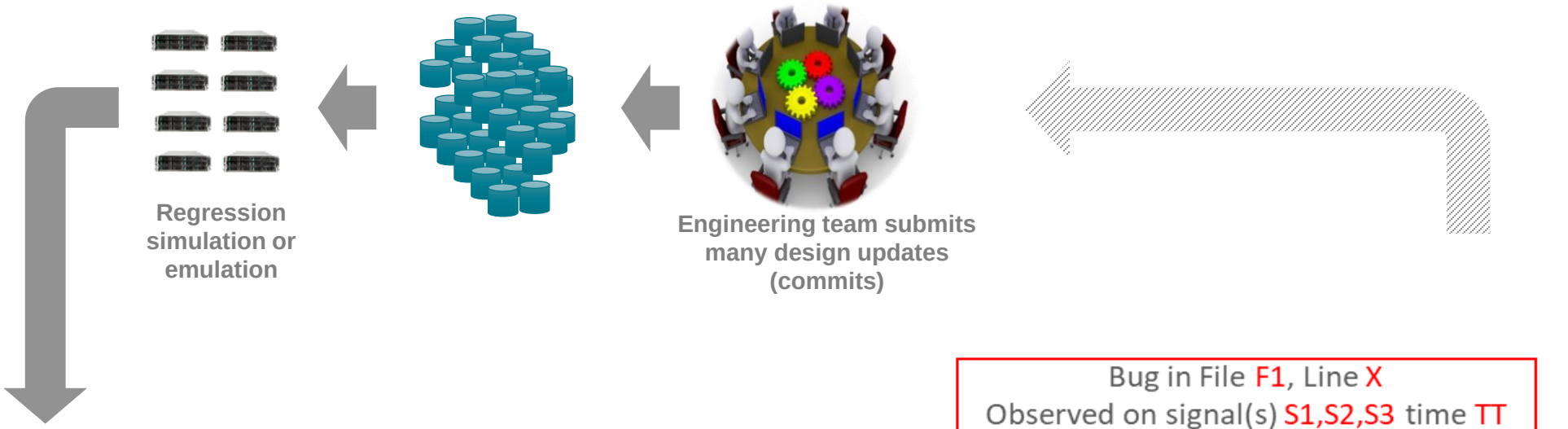
Focus compute resources on gaps in verification

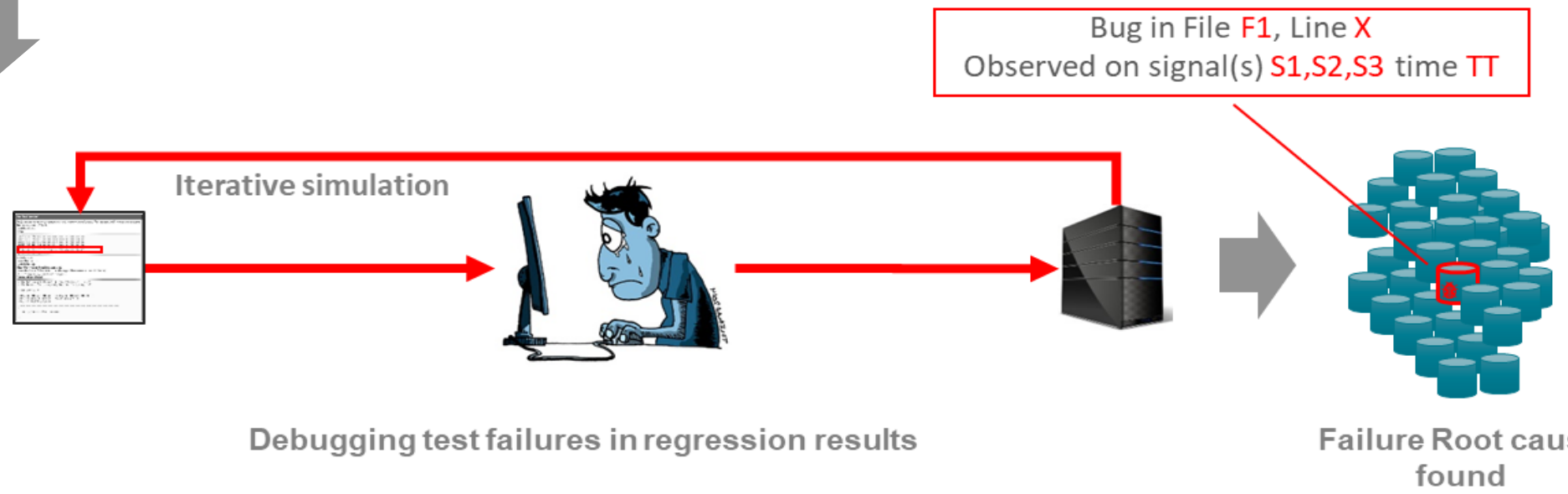
Verification Challenge – Regression Debug



- Integrating and verifying multiple (10s-100s) IPs
- Each of the IP is constantly changing, evolving, improving
- Week to week, block and system level testing results in test failures
- Determining the root cause of the failure requires multiple engineers and considerable time

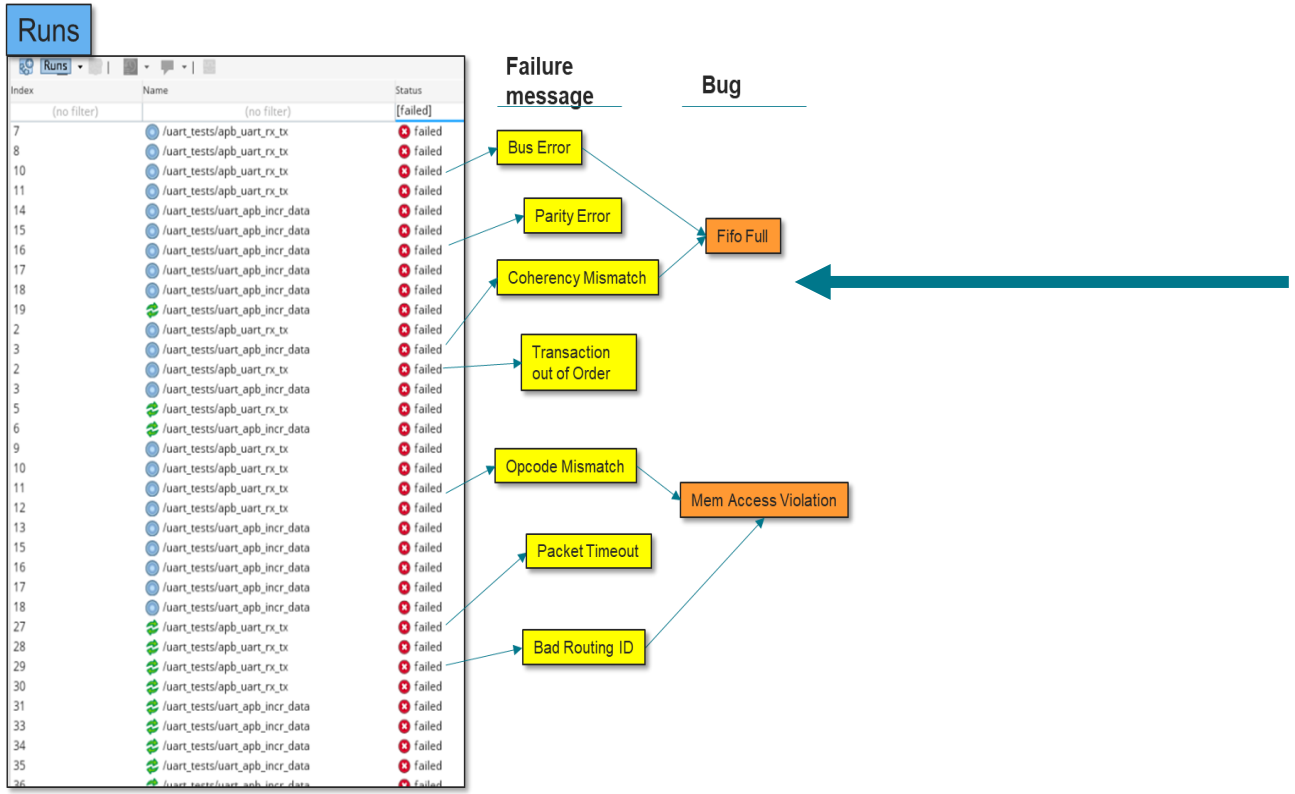
Functional Verification Debug – Traditional Process





Automated Machine Learning Bucketing of Regression Failures

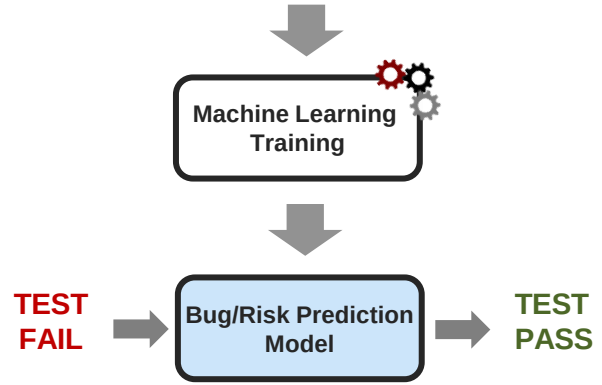
- Manual failure analysis of regression is very costly and inefficient

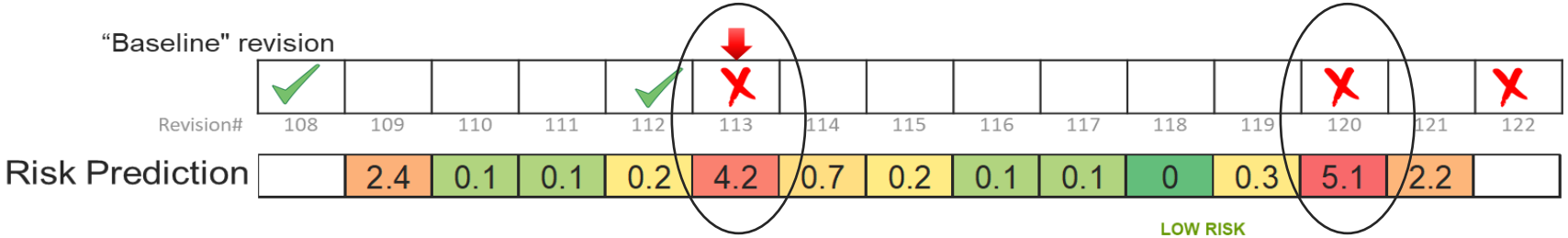


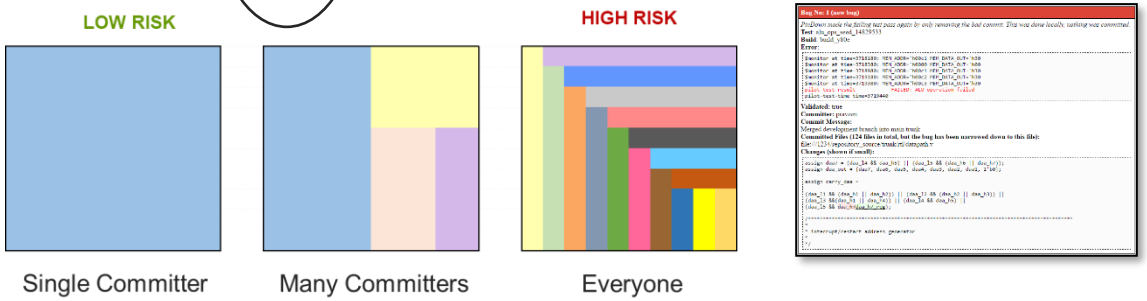
- Automated failure analysis
  - Automate failure classification using machine learning
  - Recognize patterns in any run-based attributes
    - Error message, test name, run time, etc.

Machine Learning to Identify What Caused Tests to Fail

- Identify and validate the commit that caused the failure
- Identify most likely source of failing testcase
- Automatically validates and isolates potential source issues by removing and rerunning the tests





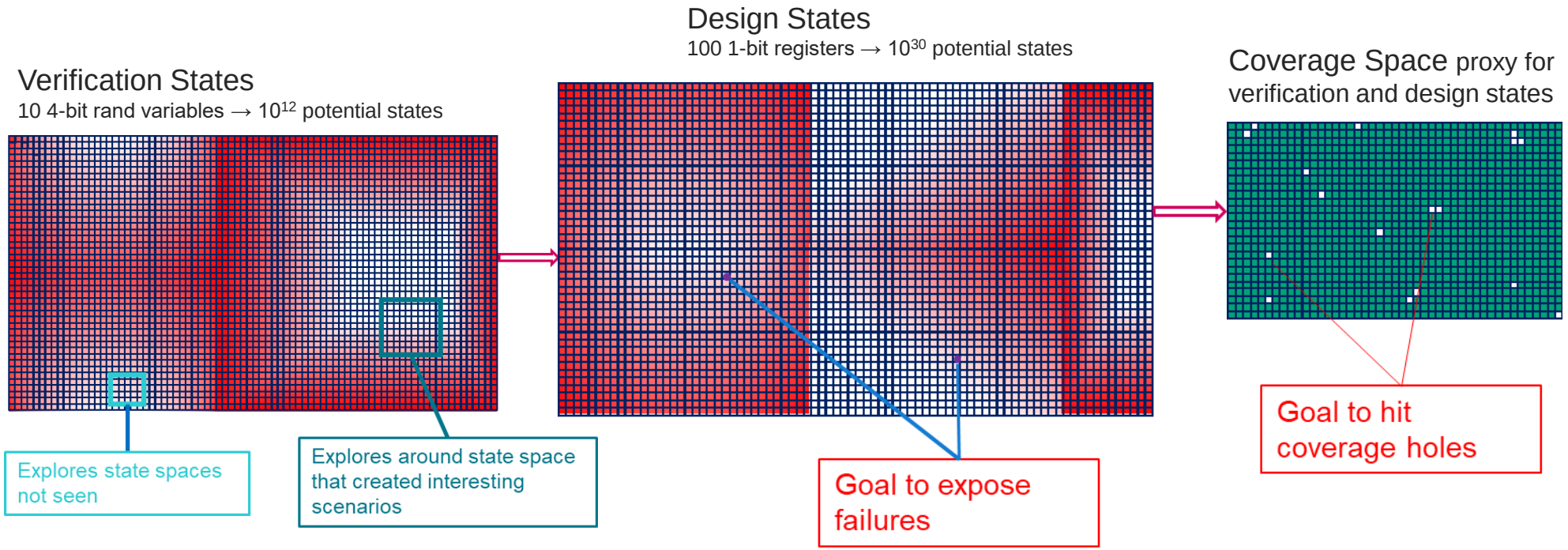


Bug Report

- Time of day, file ownership and complexity
  - More users increase the risk.
  - Certain users may have a history of bad commits

Simulation Level ML to Improve Random Verification

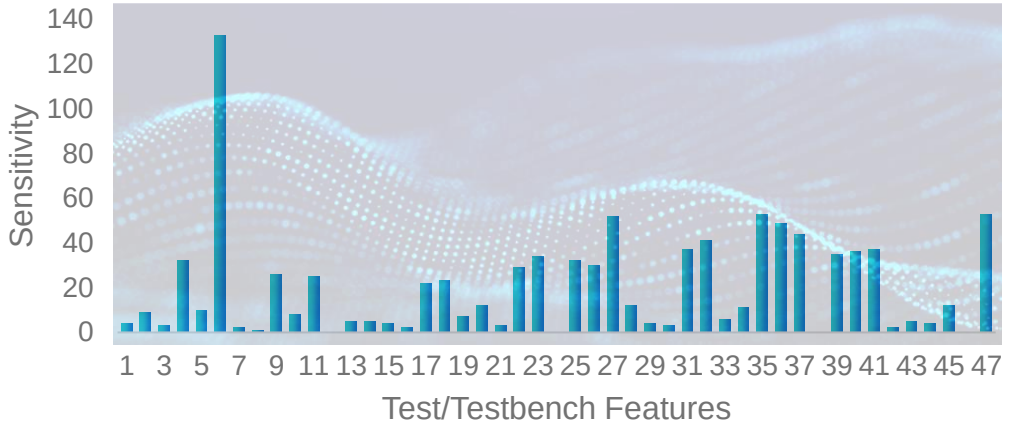
- Remove redundant runs
- Spread out the verification space exposure
  - Reduce high hit areas
  - Increase low hit areas
- Use coverage as proxy to spread out verification of interesting design states
- Use failures as an indicator of important focus points (cousin bug hunting)



ML Analytics to Expose Patterns

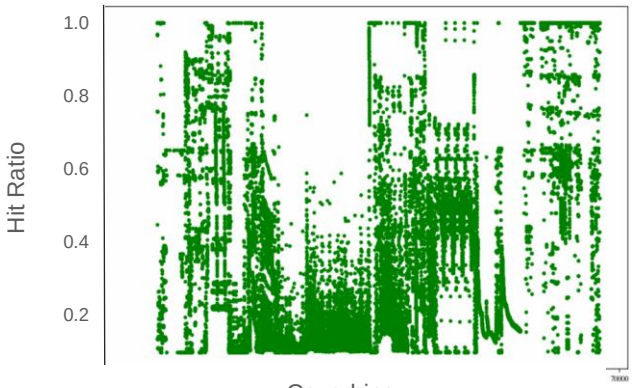
- Sensitivity analysis
- Run distribution
- Hit ratio analysis

Sensitivity

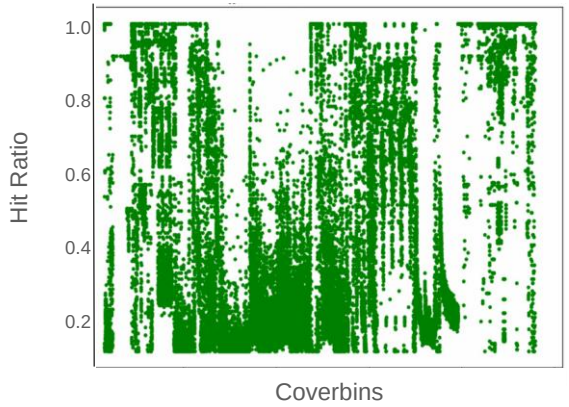


Hit ratio analysis

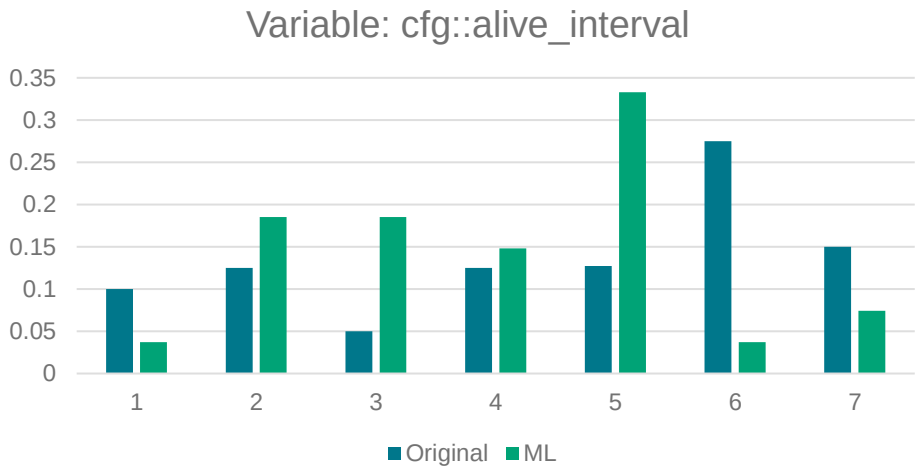
Original Regression: 10,000 CPU hrs



ML Regression: 100 CPU hrs



Random variable analysis



Where Simulation ML Can Fit in a Typical MDV Project

Environment development

- Develop / Reuse components
- Create tests
- Add functional coverage model

RTL Verification / bugfix cycle

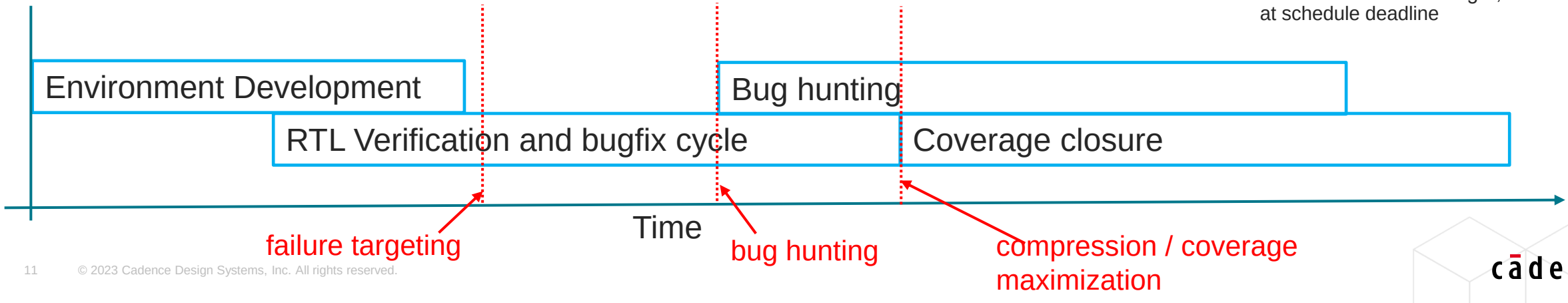
- Developer check in tests
- Nightly runs on code changes
- Weekly complete regressions
- Ends when bug rate hits a suitably low threshold

Bug hunting

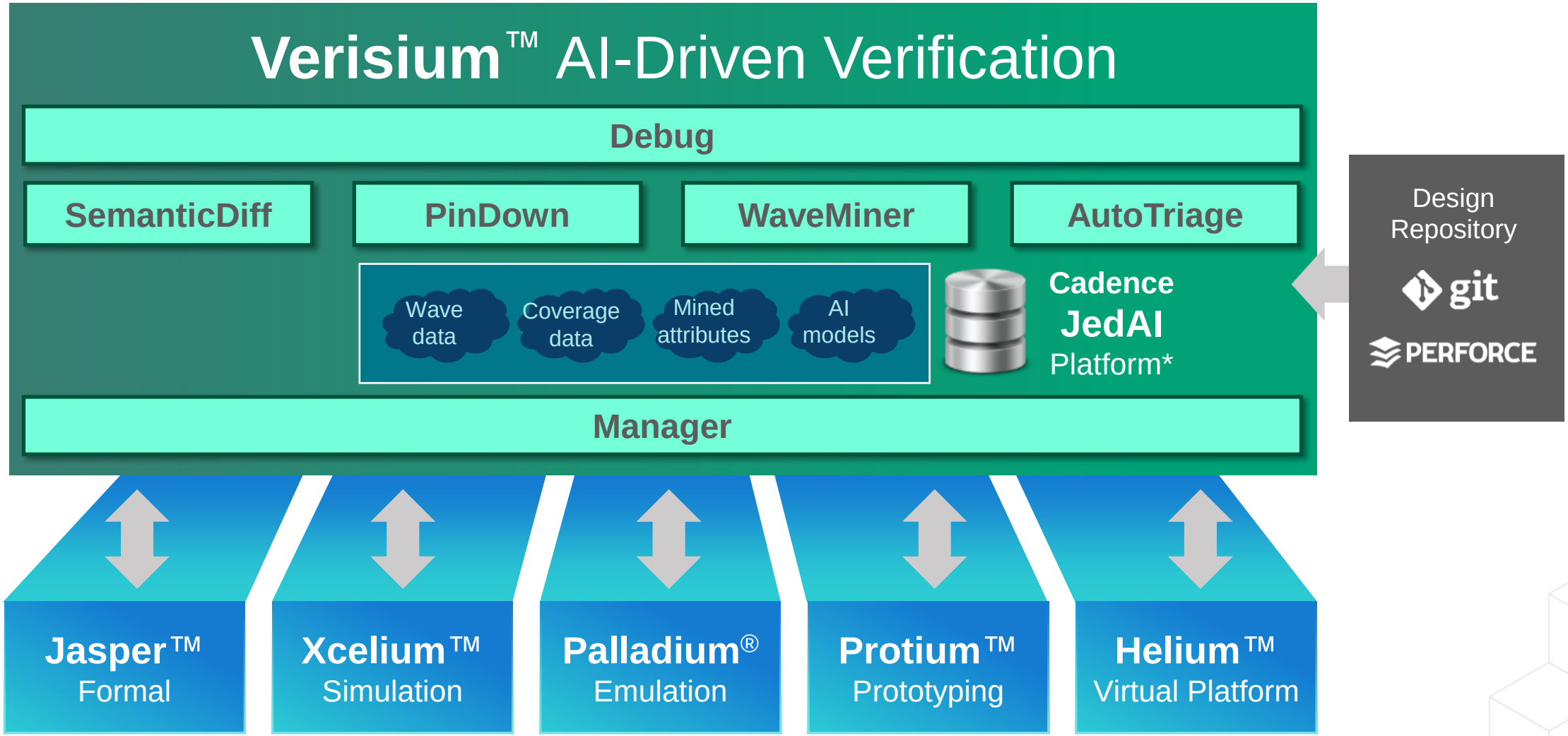
- Add additional corner case scenarios
- Fill up available resources with randomized runs
- Typically budgeted for a specific amount of time and given specific resources

Coverage closure

- Regress continuously until convergence
- Analyze gap to 100%
- Refine and create directed testcases
- Ends with 100% coverage achieved or “close enough”, i.e. at schedule deadline



Verisium AI-Driven Verification Platform



* Cadence® Joint Enterprise Data and AI (JedAI) Platform