2023
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
TAIWAN

AI Driven Verification

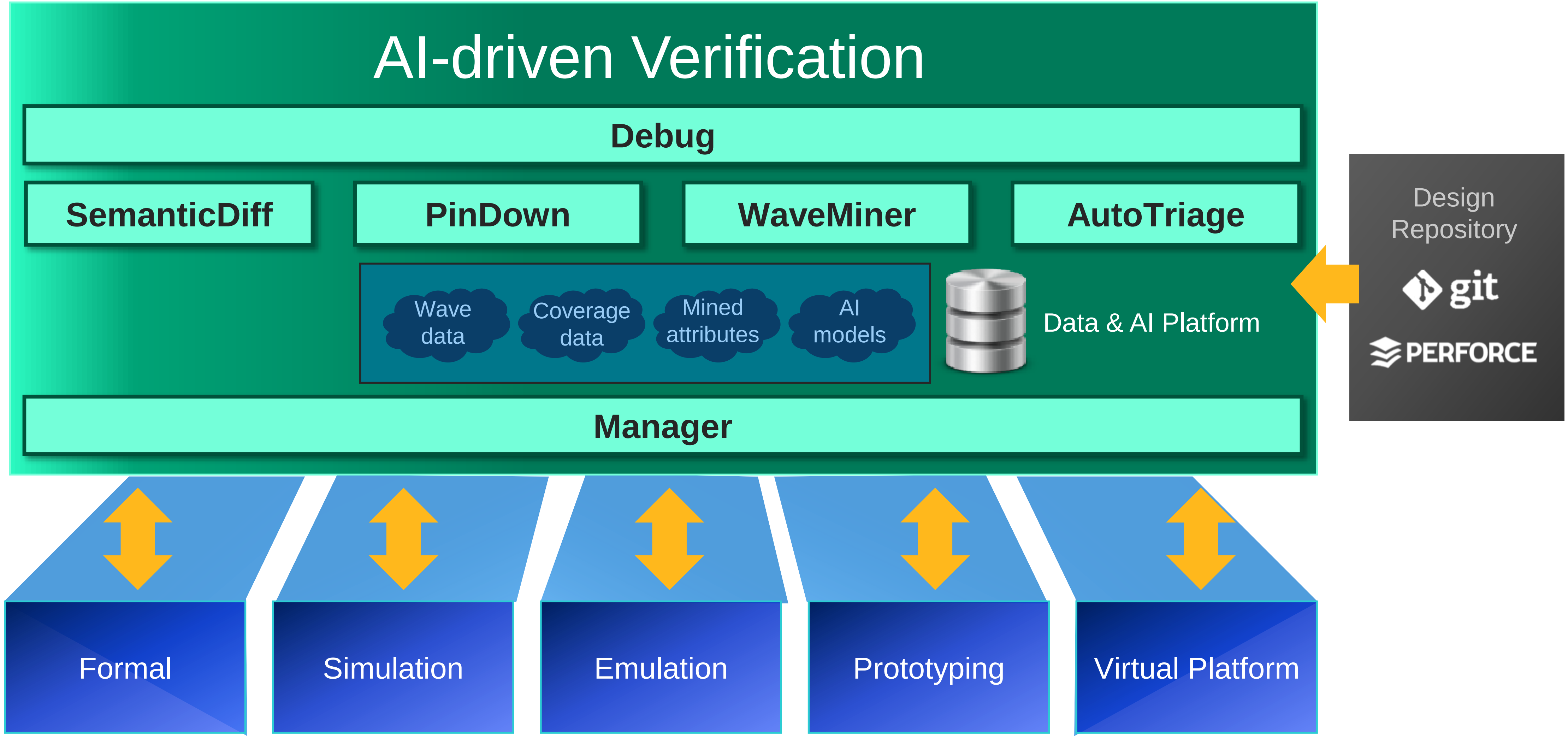
Curtis Tsai

Cadence Design Systems

cadence®

accellera
SYSTEMS INITIATIVE

Introduction



AI-driven Verification

Debug

SemanticDiff

PinDown

WaveMiner

AutoTriage

Wave data

Coverage data

Mined attributes

AI models

Data & AI Platform

Manager

Design Repository

git

PERFORCE

Formal

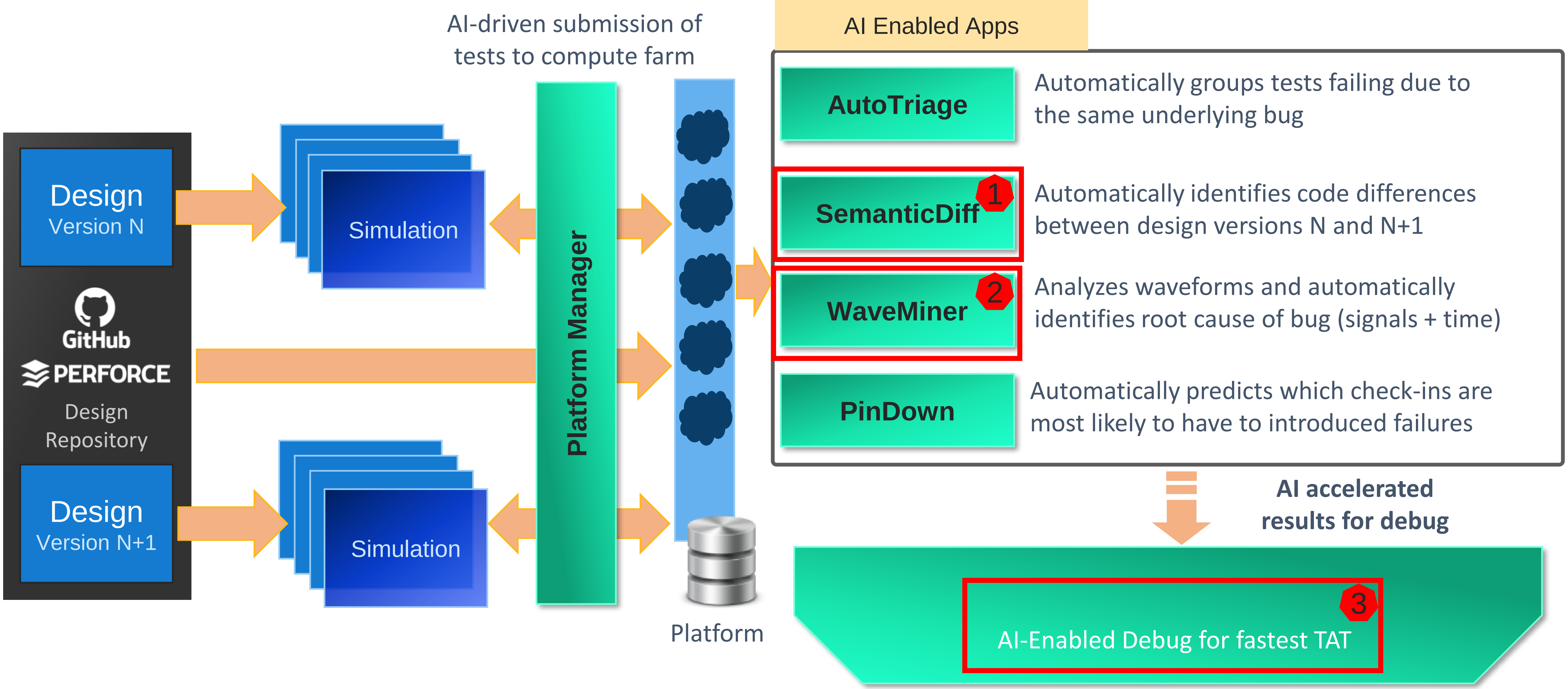
Simulation

Emulation

Prototyping

Virtual Platform

Verification Platform for Fastest Debug TAT



AI-driven submission of tests to compute farm

Design Version N

GitHub

PERFORCE

Design Repository

Design Version N+1

Simulation

Simulation

Platform Manager

Platform

AI Enabled Apps

AutoTriage	Automatically groups tests failing due to the same underlying bug
SemanticDiff 1	Automatically identifies code differences between design versions N and N+1
WaveMiner 2	Analyzes waveforms and automatically identifies root cause of bug (signals + time)
PinDown	Automatically predicts which check-ins are most likely to have to introduced failures

AI accelerated results for debug

3 AI-Enabled Debug for fastest TAT

What is Semantic Diff ?

- An advanced RTL design comparison tool that compares the two versions of RTL design and determines the Semantic Differences between them

- More sophisticated than text-based diff
  - **It ignores comments/blank spaces/newlines**
  - **It improves productivity as user can concentrate on files with maximum code changes**

- It reports the entities and other details of the RTL design with a specific rank where the Semantic Differences, user can concentrate on RTL where rank is higher which implies there are more Semantic Diff's

Semantic Diff Flow

Identify and rank semantic changes between two RTL versions

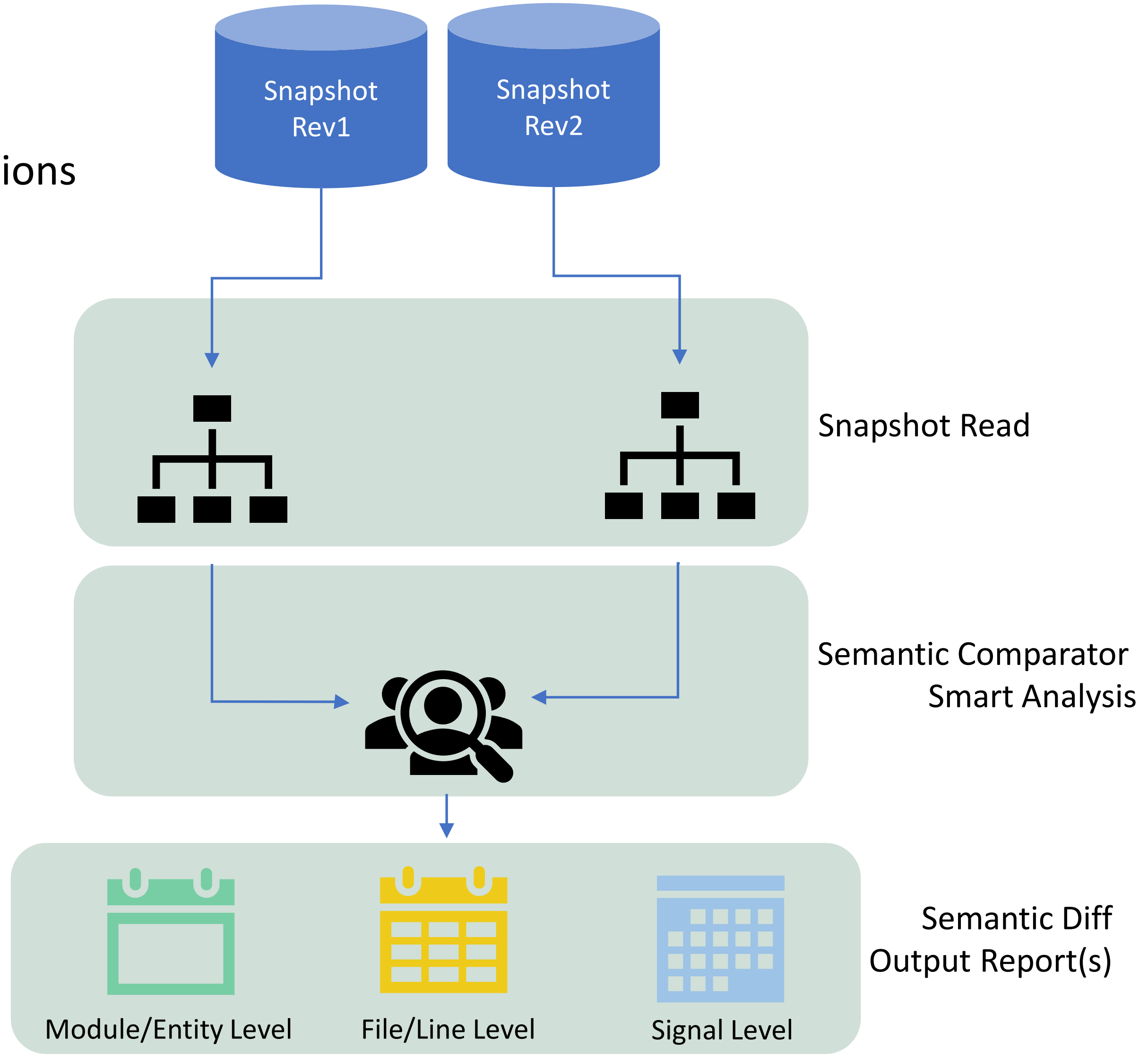
- Ignore harmless changes | Rank “complexity” of genuine logic changes

```
module cg (d, clk);
 input d, clk;
 reg orig;
 reg clone;
 reg g_latch;
 wire w = orig ^ d;
 wire gclk = clk & g_latch;

 always @(clk or w)
    if (~clk) g_latch <= w;

 always @(posedge gclk)  clone <= d;

 always @(posedge clone) orig <= d;

 fd : assert property (
        @(posedge clk) orig == clone
 );
Endmodule
```

?

=

```
module cg (d, clk);
 input d, clk;
 reg orig, clone, g_latch;

 // Comments ...
 wire w = orig ^ d;
 wire gclk = clk & g_latch;

 always @(clk or w)
    if (clk) g_latch <= w;

 always @(posedge gclk)
      clone <= d;

 always @(posedge clone)
      orig <= d;

 fd : assert property (
        @(posedge clk) orig == clone
 );
endmodule
```

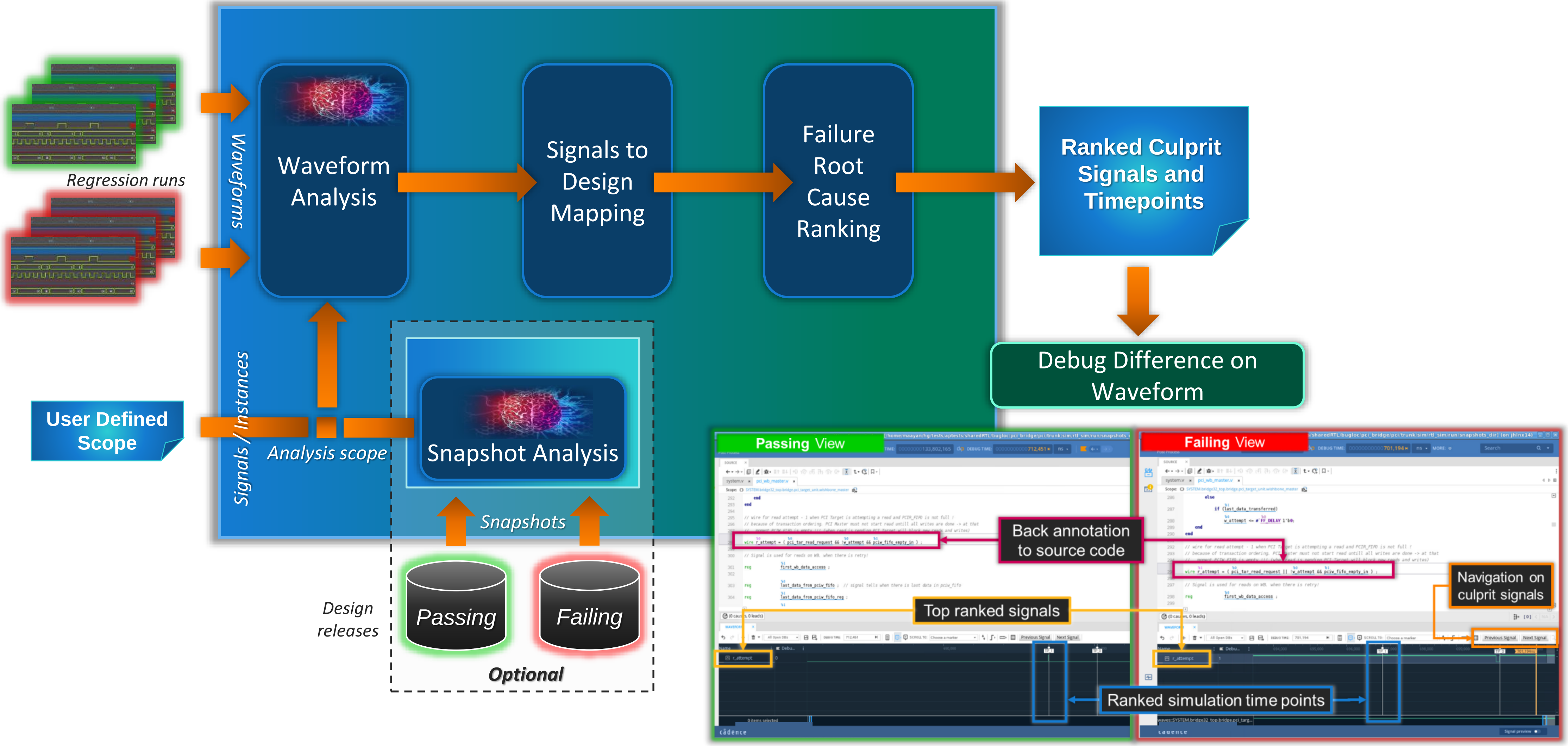


Snapshot Read

Semantic Comparator
Smart Analysis

Semantic Diff
Output Report(s)

WaveMiner flow



Regression runs

Waveforms

Waveform Analysis

Signals to Design Mapping

Failure Root Cause Ranking

Ranked Culprit Signals and Timepoints

Debug Difference on Waveform

User Defined Scope

Signals / Instances

Analysis scope

Snapshot Analysis

Snapshots

Design releases

Passing

Failing

Optional

Passing View

/home/maayan/hg/tests/aptests/sharedRTL/bugloc/pci_bridge/pci/trunk/sim/rtl_sim/run/snapshots_

TIME: 00000000133,802,165 DEBUG TIME: 00000000000712,451 ns

system.v pci_wb_master.v

Scope: SYSTEM.bridge32_top.bridge.pci_target_unit.wishbone_master

```
        end
    end

    // wire for read attempt - 1 when PCI Target is attempting a read and PCIR_FIFO is not full !
    // because of transaction ordering, PCI Master must not start read untill all writes are done -> at that
    // moment PCIW_FIFO is empty !!! (when read is pending PCI Target will block new reads and writes)
    wire r_attempt = ( pci_tar_read_request && !w_attempt && pciw_fifo_empty_in ) ;

    // Signal is used for reads on WB, when there is retry!
    reg             first_wb_data_access ;

    reg             last_data_from_pciw_fifo ;  // signal tells when there is last data in pciw_fifo
    reg             last_data_from_pciw_fifo_reg ;
```

Failing View

/sharedRTL/bugloc/pci_bridge/pci/trunk/sim/rtl_sim/run/snapshots_dir] (on jhlnx14)

DEBUG TIME: 00000000000701,194 ns MORE:

system.v pci_wb_master.v

Scope: SYSTEM.bridge32_top.bridge.pci_target_unit.wishbone_master

```
            else
                if (last_data_transferred)
                    w_attempt <= #`FF_DELAY 1'b0;
            end
    end

    // wire for read attempt - 1 when PCI Target is attempting a read and PCIR_FIFO is not full !
    // because of transaction ordering, PCI Master must not start read untill all writes are done -> at that
    // moment PCIW_FIFO is empty !!! (when read is pending PCI Target will block new reads and writes)
    wire r_attempt = ( pci_tar_read_request || !w_attempt && pciw_fifo_empty_in ) ;

    // Signal is used for reads on WB, when there is retry!
    reg             first_wb_data_access ;
```

Back annotation to source code

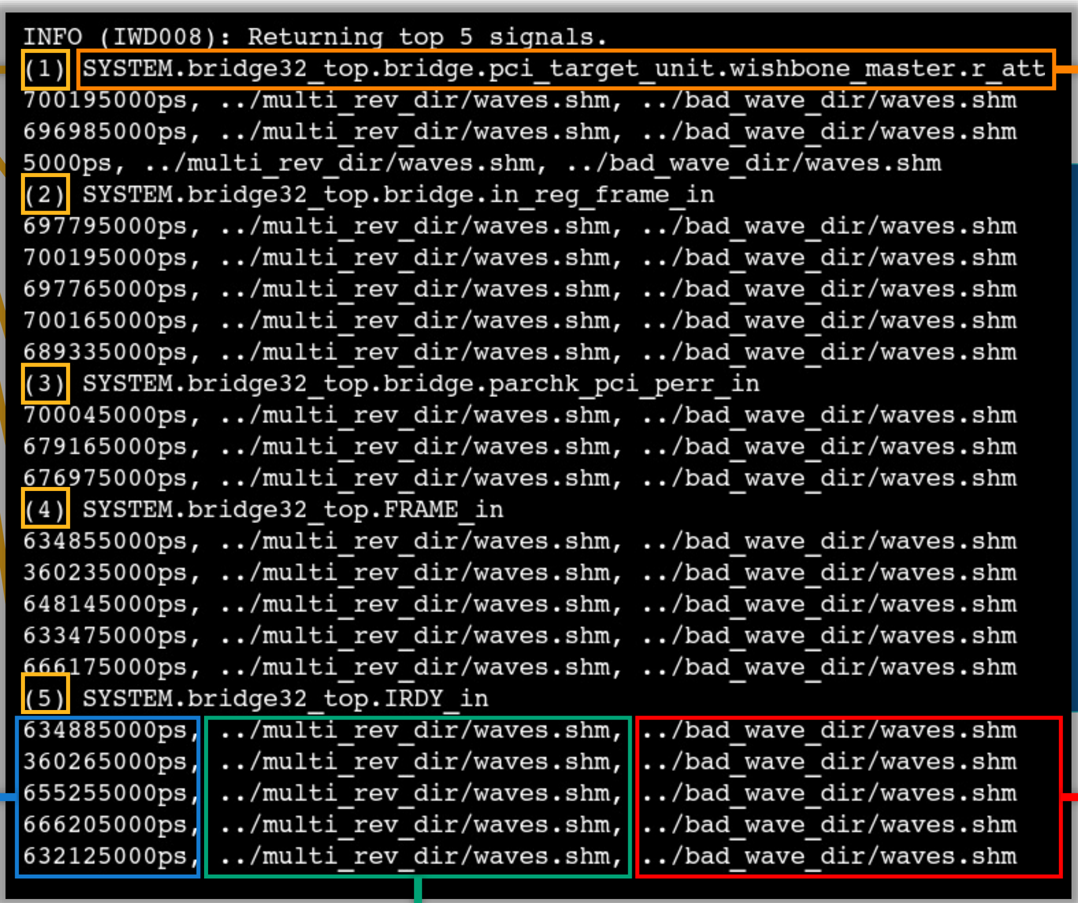
Top ranked signals

Navigation on culprit signals

Ranked simulation time points

WaveMiner Results

Text Report



```
INFO (IWD008): Returning top 5 signals.
(1) SYSTEM.bridge32_top.bridge.pci_target_unit.wishbone_master.r_att
700195000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
696985000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
5000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
(2) SYSTEM.bridge32_top.bridge.in_reg_frame_in
697795000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
700195000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
697765000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
700165000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
689335000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
(3) SYSTEM.bridge32_top.bridge.parchk_pci_perr_in
700045000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
679165000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
676975000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
(4) SYSTEM.bridge32_top.FRAME_in
634855000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
360235000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
648145000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
633475000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
666175000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
(5) SYSTEM.bridge32_top.IRDY_in
634885000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
360265000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
655255000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
666205000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
632125000ps, ../multi_rev_dir/waves.shm, ../bad_wave_dir/waves.shm
```

Signals ranked by relevance

Full hierarchical signal name

Top <N> ranked culprit signals and simulation timepoints to be investigated by the user

Ranked timepoints per signal

Failing waveform location

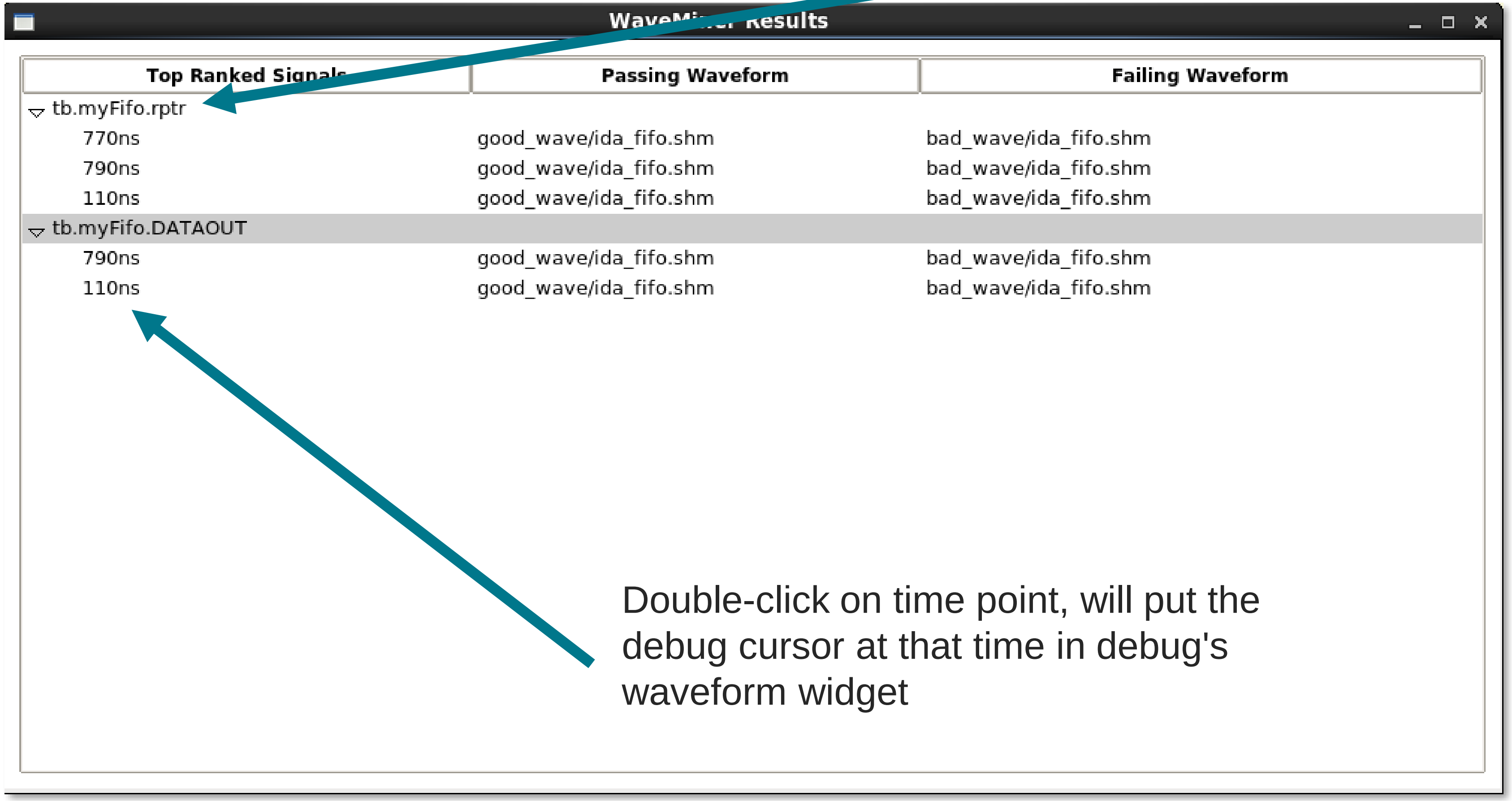
Passing waveform location

WaveMiner Results Widget

New widget enables easy navigation on the results and provides high level information

Double-click on signal, will bring it to waveform widget

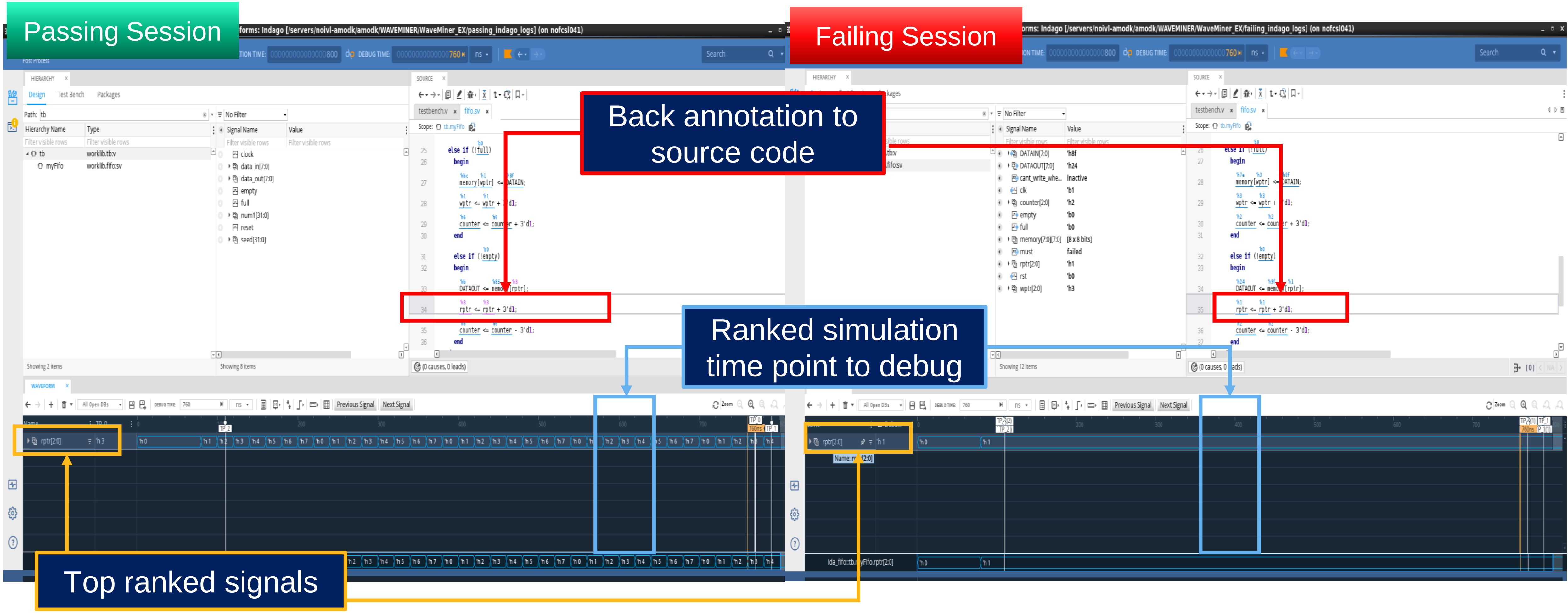
Signals are ranked in decreasing relevance order



Top Ranked Signals	Passing Waveform	Failing Waveform
▽ tb.myFifo.rptr		
770ns	good_wave/ida_fifo.shm	bad_wave/ida_fifo.shm
790ns	good_wave/ida_fifo.shm	bad_wave/ida_fifo.shm
110ns	good_wave/ida_fifo.shm	bad_wave/ida_fifo.shm
▽ tb.myFifo.DATAOUT		
790ns	good_wave/ida_fifo.shm	bad_wave/ida_fifo.shm
110ns	good_wave/ida_fifo.shm	bad_wave/ida_fifo.shm

Double-click on time point, will put the debug cursor at that time in debug's waveform widget

WaveMiner - Visualization



Passing Session

Failing Session

Back annotation to source code

Ranked simulation time point to debug

Top ranked signals

forms: Indago [/servers/noivl-amodk/amodk/WAVEMINER/WaveMiner_EX/passing_indago_logs] (on nofcsl041)

orms: Indago [/servers/noivl-amodk/amodk/WAVEMINER/WaveMiner_EX/failing_indago_logs] (on nofcsl041)

Hierarchy Name	Type
tb	worklib.tb.v
myFifo	worklib.fifo.sv

Signal Name (passing): clock, data_in[7:0], data_out[7:0], empty, full, num1[31:0], reset, seed[31:0]

Signal Name	Value
DATAIN[7:0]	'h8f
DATAOUT[7:0]	'h24
cant_write_whe...	inactive
clk	'b1
counter[2:0]	'h2
empty	'b0
full	'b0
memory[7:0][7:0]	[8 x 8 bits]
must	failed
rptr[2:0]	'h1
rst	'b0
wptr[2:0]	'h3

Scope: tb.myFifo (fifo.sv)

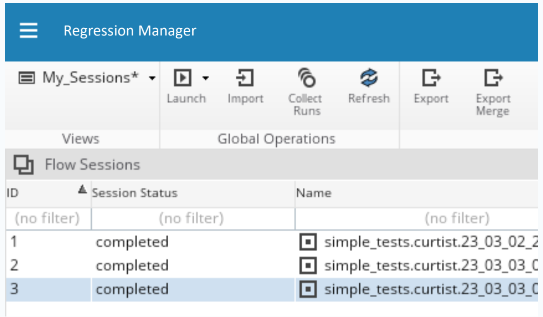
```
  else if (!full)
    begin
      memory[wptr] <= DATAIN;
      wptr <= wptr + 3'd1;
      counter <= counter + 3'd1;
    end

    else if (!empty)
    begin
      DATAOUT <= memory[rptr];
      rptr <= rptr + 3'd1;
      counter <= counter - 3'd1;
    end
```

Now you have the two Debug sessions and signal is there, with markers for its time points. The recommendation is to do driver tracing at the time points suggested by WaveMiner

AutoFocus flow

Regression Data

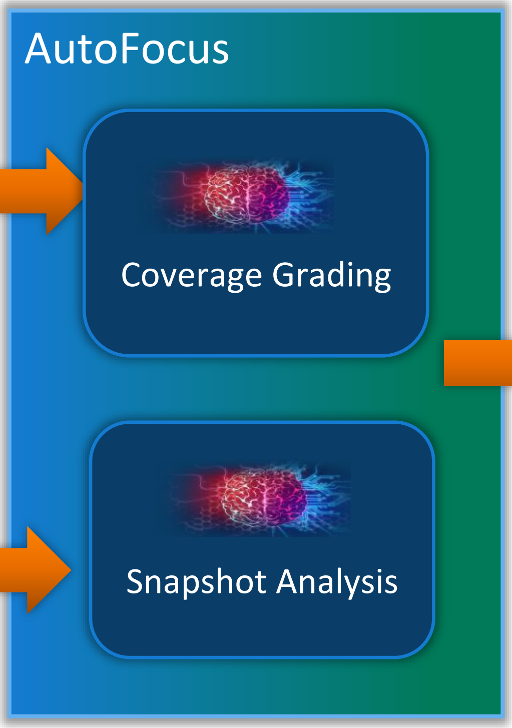
Test_case1
Test_case2
....
Test_caseN



ID	Session Status	Name
(no filter)	(no filter)	(no filter)
1	completed	simple_tests.curtist.23_03_02_2
2	completed	simple_tests.curtist.23_03_03_C
3	completed	simple_tests.curtist.23_03_03_C

New Snapshot

Old Snapshot



Output (JSON format)

Test_case1, seed=32121134
Test_case5, seed=23422324
……

The output format includes

- Test case name
- The corresponding random seed

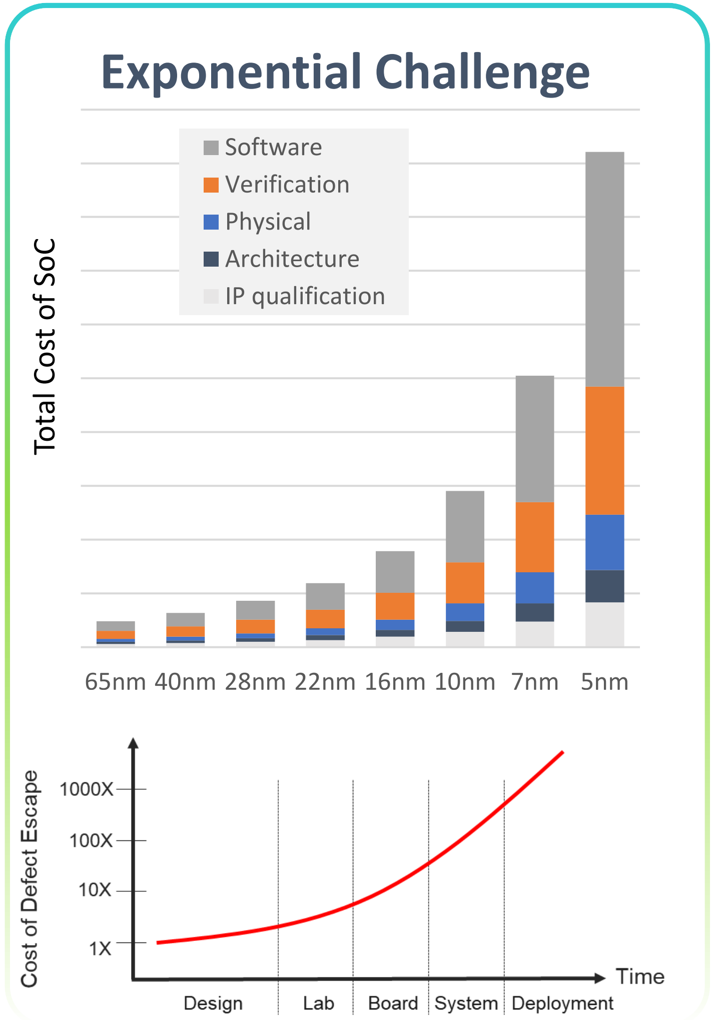
Target

- Hit 90% of original regression coverage for modified modules

AI for Regression Productivity

Machine Learning for coverage closure

Trends in Hardware / Software Development

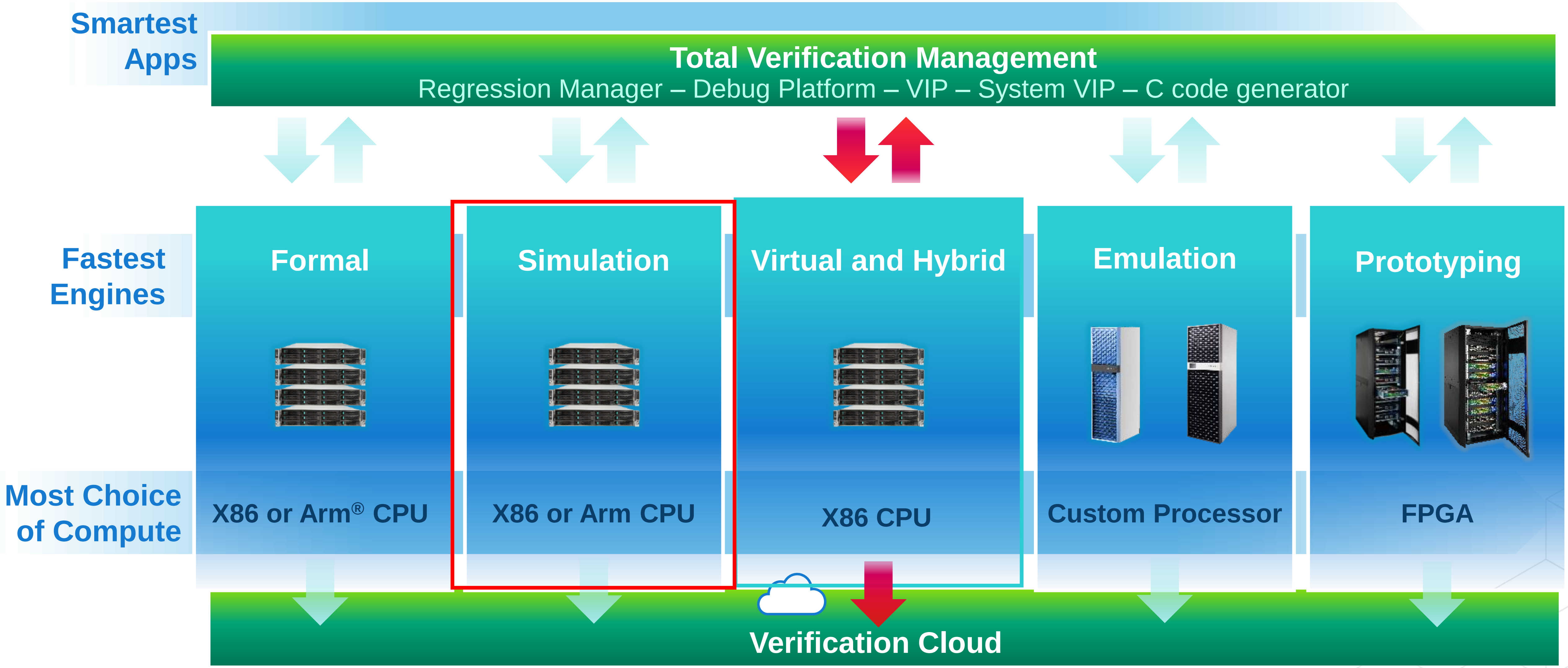


ROI mindset: Bugs found per $ per day

Verification Throughput

Verification Solution

Find and fix the most bugs per $ compute per day



Smartest Apps

Total Verification Management

Regression Manager – Debug Platform – VIP – System VIP – C code generator

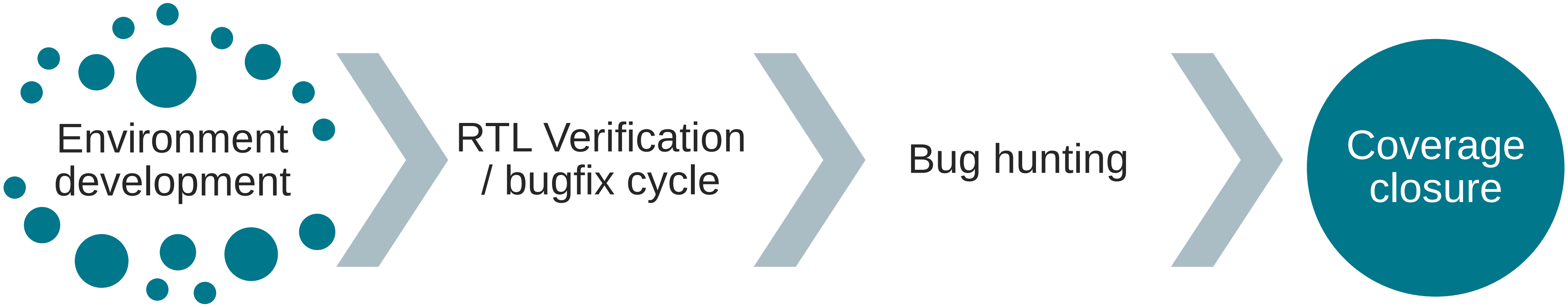
Fastest Engines

Formal	Simulation	Virtual and Hybrid	Emulation	Prototyping
X86 or Arm® CPU	X86 or Arm CPU	X86 CPU	Custom Processor	FPGA

Most Choice of Compute

Verification Cloud

Where Does Machine Learning Fit in a Typical MDV Timeline



Environment development

RTL Verification / bugfix cycle

Bug hunting

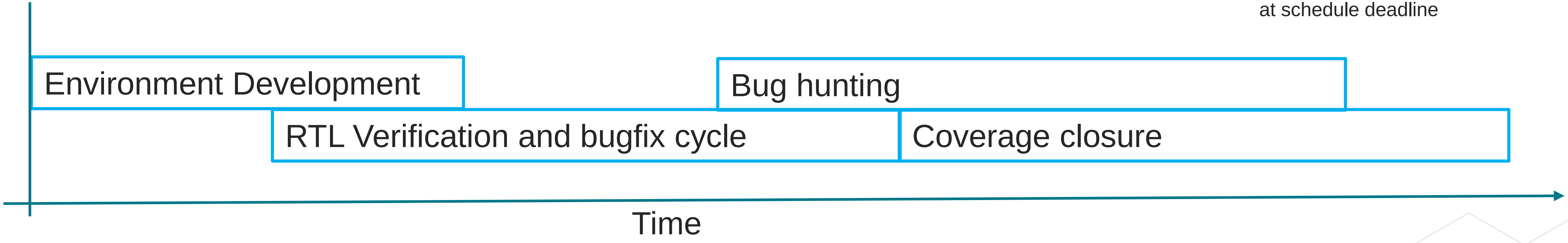
Coverage closure

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

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

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

- 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



Environment Development

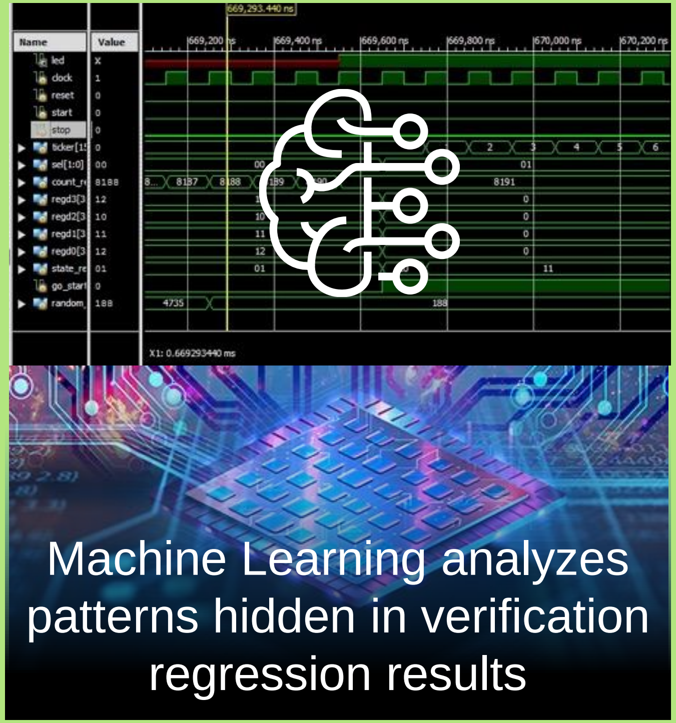
Bug hunting

RTL Verification and bugfix cycle

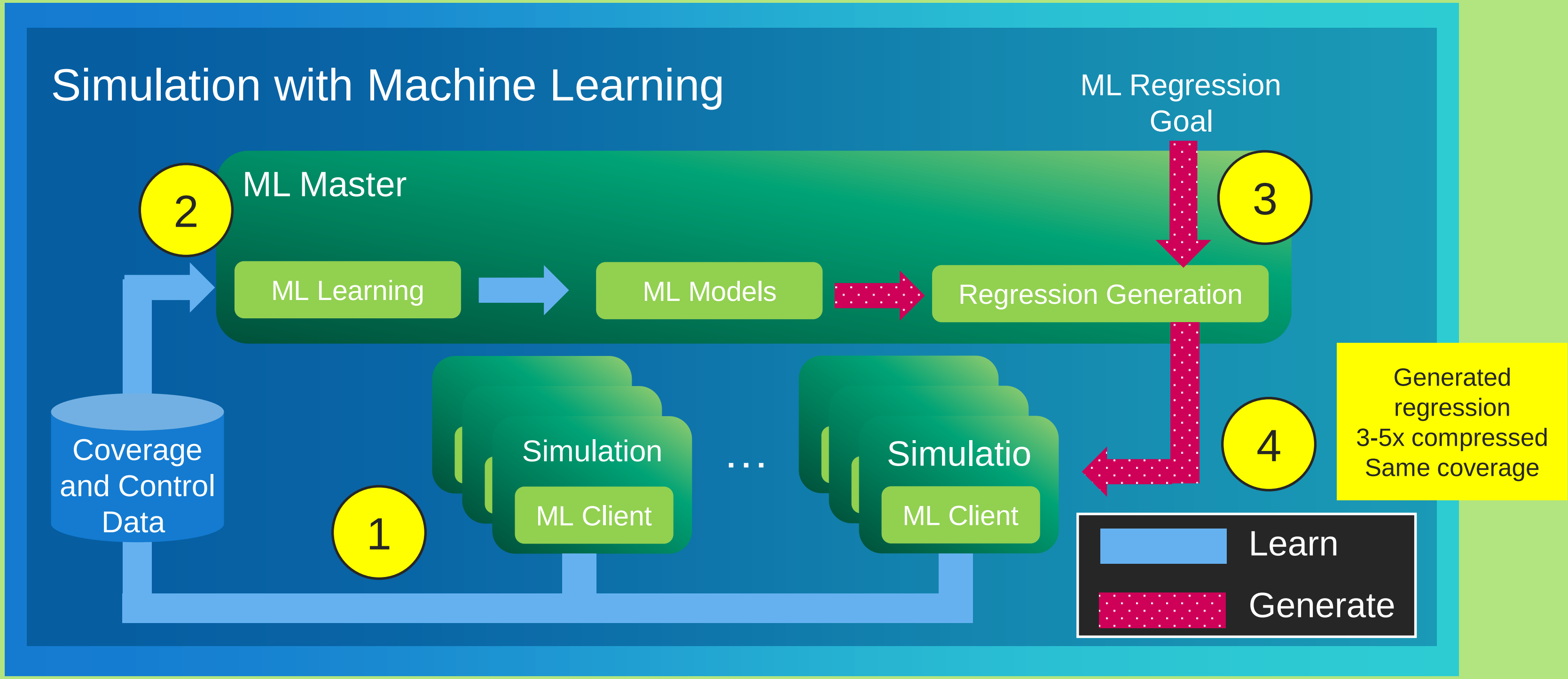
Coverage closure

Time

The Machine Learning Flow



Machine Learning analyzes patterns hidden in verification regression results



Original Regression

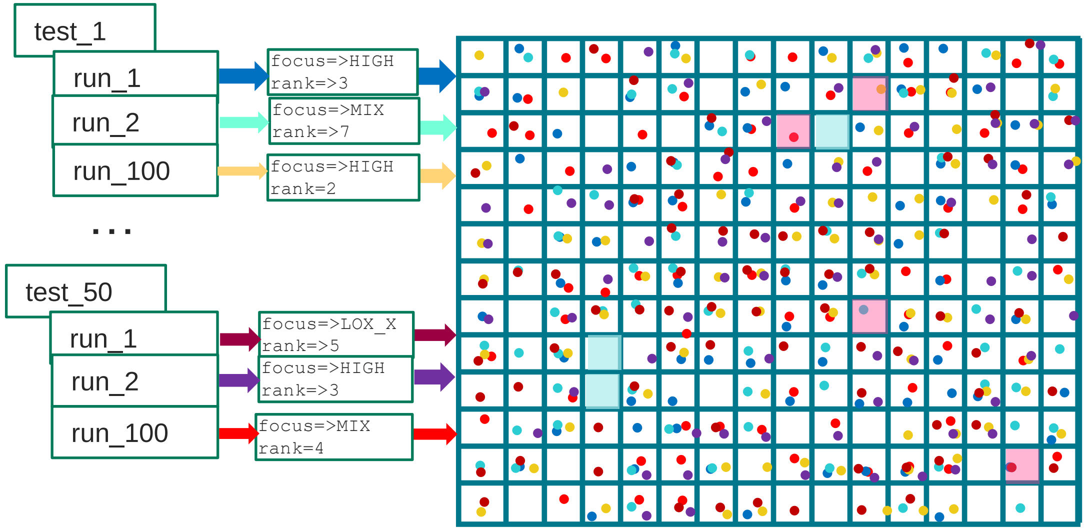
- 50 tests, 100 seeds per test (5,000 runs)

Random control

```
class cfg_c extends uvm_sequence_item;
  rand focus_e focus;
  rand [2:0] rank;
  ...
endclass
```

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

Original Regression Coverage model



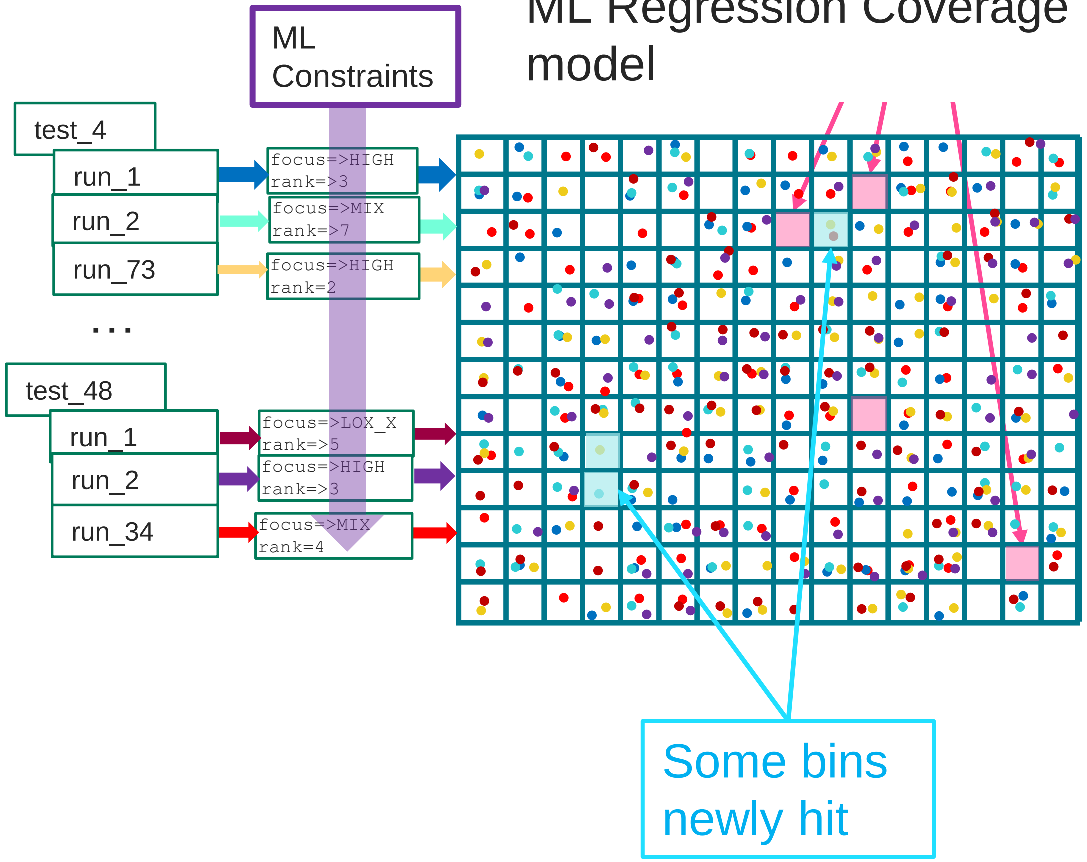
Generate new regression runs

Generated Regression

30 tests, 1,500 runs

Some bins not regained

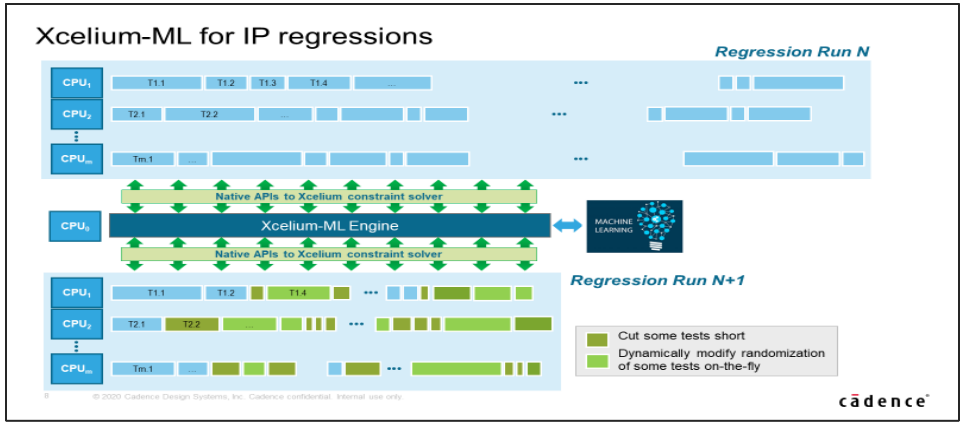
ML Regression Coverage model



Some bins newly hit

Xcelium Machine Learning App

Use Cases



Xcelium-ML for IP regressions

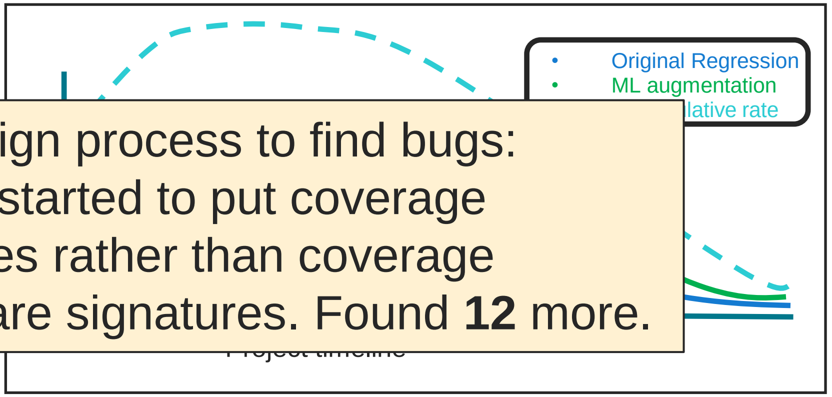
Regression Run N
CPU₁
T1.1
T1.2
T1.3
T1.4
CPU₂
T2.1
T2.2
CPUₘ
Tm.1
Native APIs to Xcelium constraint solver
CPU₀
Xcelium-ML Engine
MACHINE LEARNING
Native APIs to Xcelium constraint solver
Regression Run N+1
CPU₁
T1.1
T1.2
T1.4
CPU₂
T2.1
T2.2
CPUₘ
Tm.1
Cut some tests short
Dynamically modify randomization of some tests on-the-fly
8
© 2020 Cadence Design Systems, Inc. Cadence confidential. Internal use only.
cādence

Regression Compression - Same coverage in less time

Replace Original regression with ML generated regression

Can be used early in design process to find bugs:

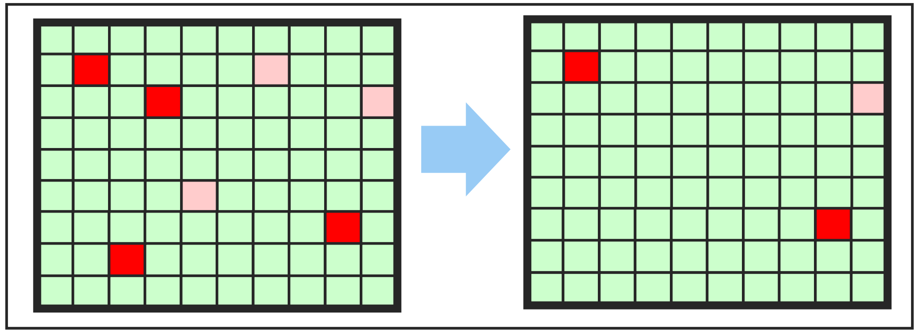
- Before even user has started to put coverage
- Target failure signatures rather than coverage
- Recent example: 17 rare signatures. Found **12** more.



Bug Hunting – Find bugs early

Augment Original regression with ML generated regression

Extend to Cousin Bug Hunting



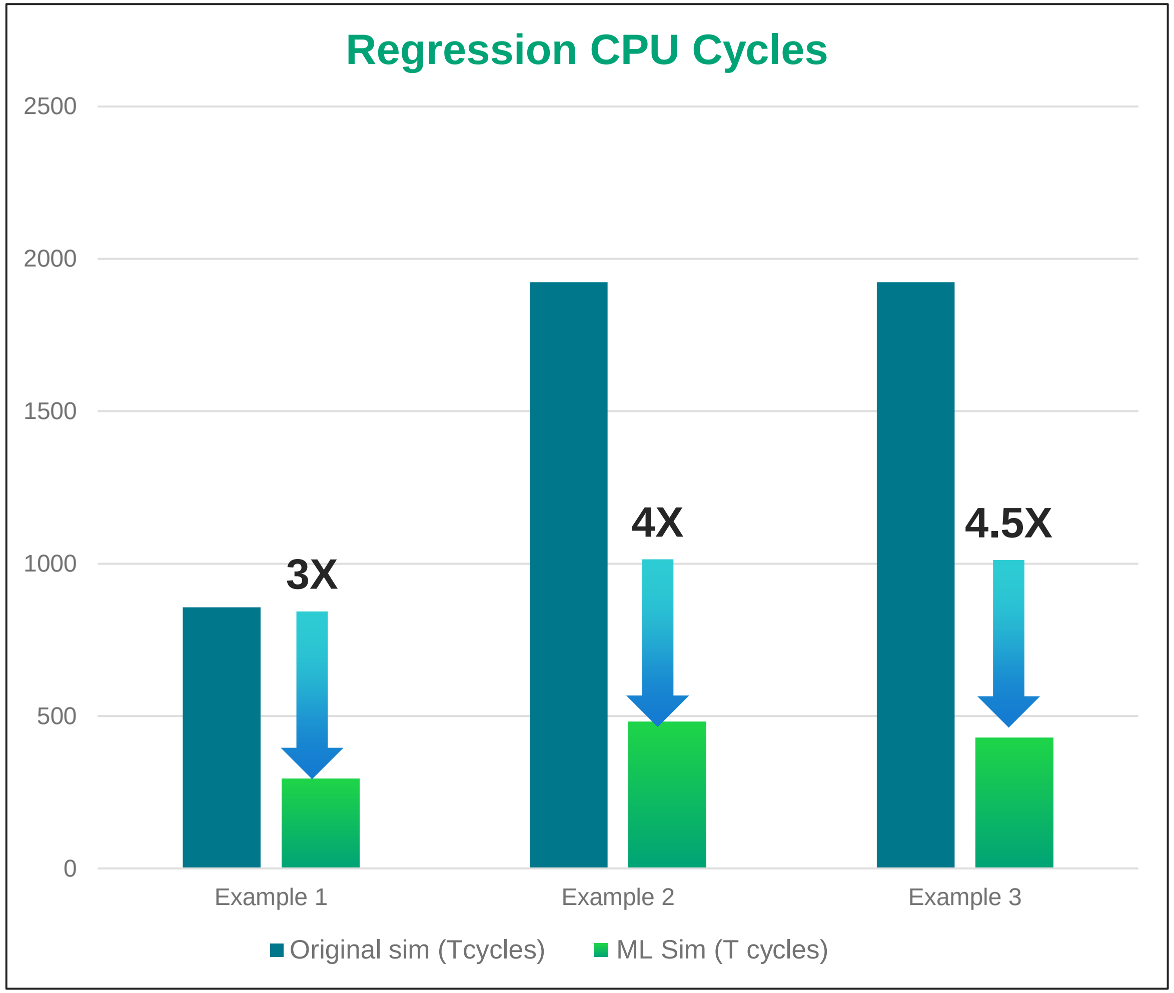
Requires a modified form of Reinforcement Learning:

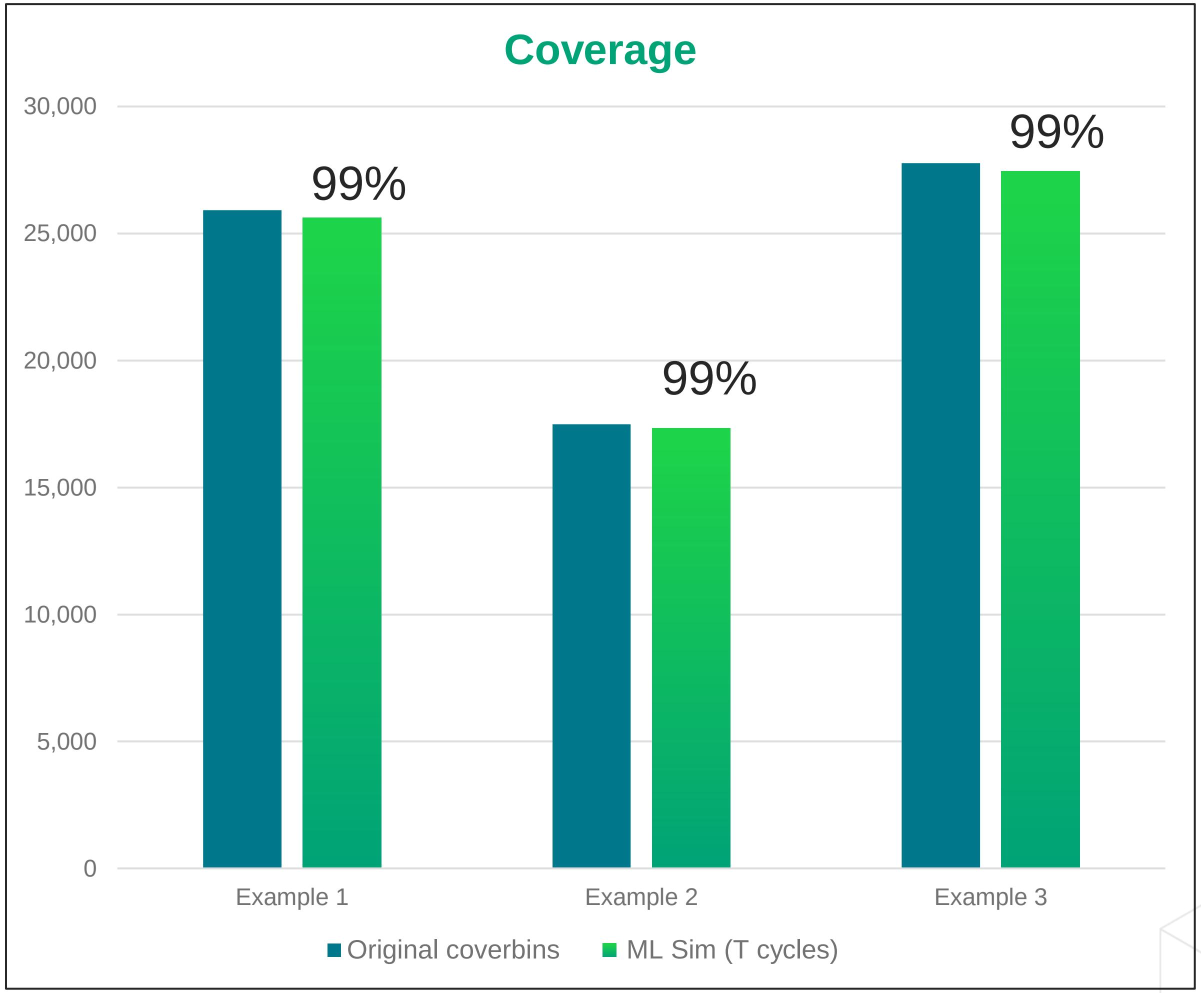
- Target unhit cross bins: cover (A x B)
- Target unhit bins using structural and statistical correlation
- Create new streams by stitching sequences

l to Coverage Maximization

to hit coverage holes

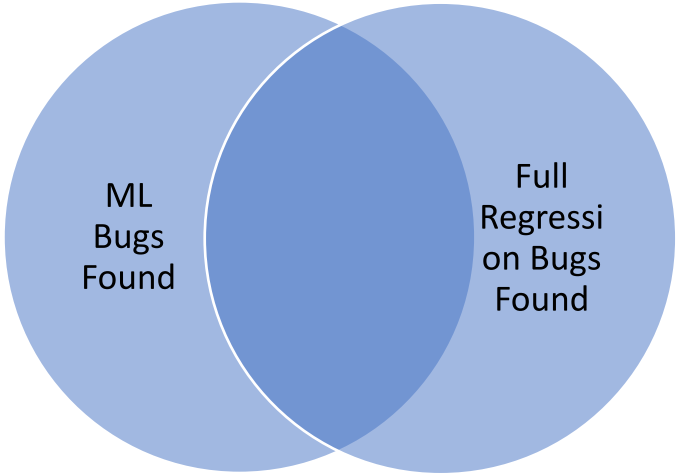
Results – Faster Regression and Matching Coverage

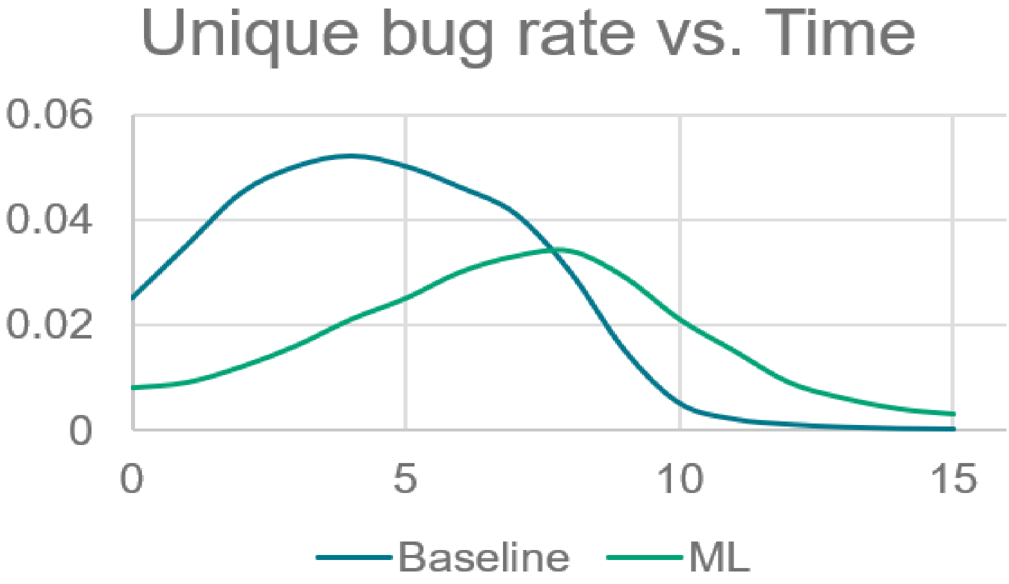




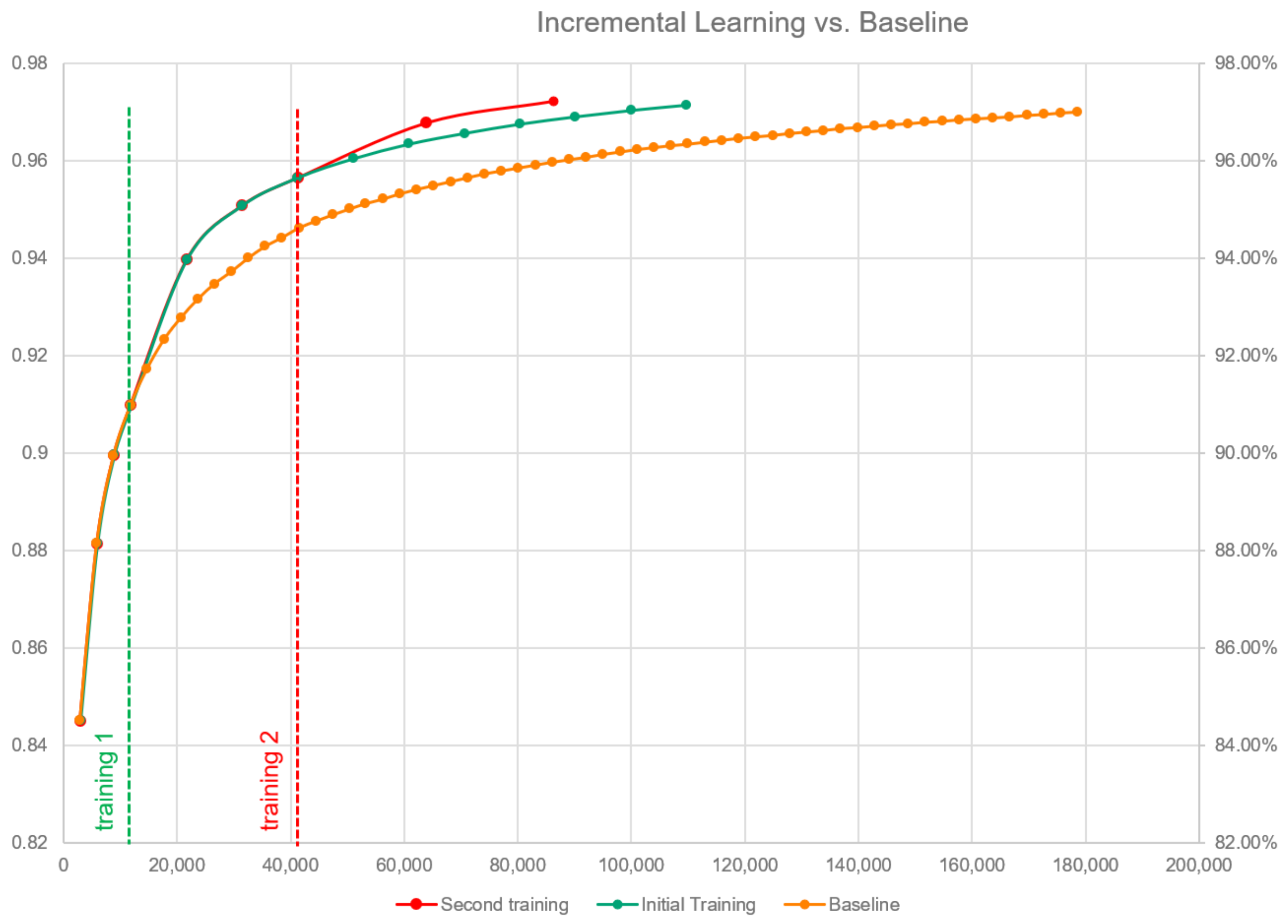
Using ML for Bug Hunting

- Augment full regression with ML-generated runs
  - The ML-generated regression will create a 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





Coverage Closure With Iterative Learning



- Orange is the baseline is regression runs without ML
- Green trains a model after 4 iterations of orange and then continues
- Red does iterative learning after 4 more iterations

2023

DESIGN AND VERIFICATION™

DVCON

CONFERENCE AND EXHIBITION

TAIWAN

accellera

SYSTEMS INITIATIVE