



Digital Verification

Budi Murty
Aug 1, 2023

Introduction



Budi has been working with Cadence since 2006 started as an Application Engineer. He has wide experience on verification technologies along with the Cadence platforms like – Xcelium, vManager and Safety. He has worked in closed collaboration with different customers around the globe to optimize the verification cycle in terms of regression throughput, automation, coverage closure, etc. Currently, he is working as Sr. AE Manager managing AE team working on different Cadence verification tools.



arm
Education



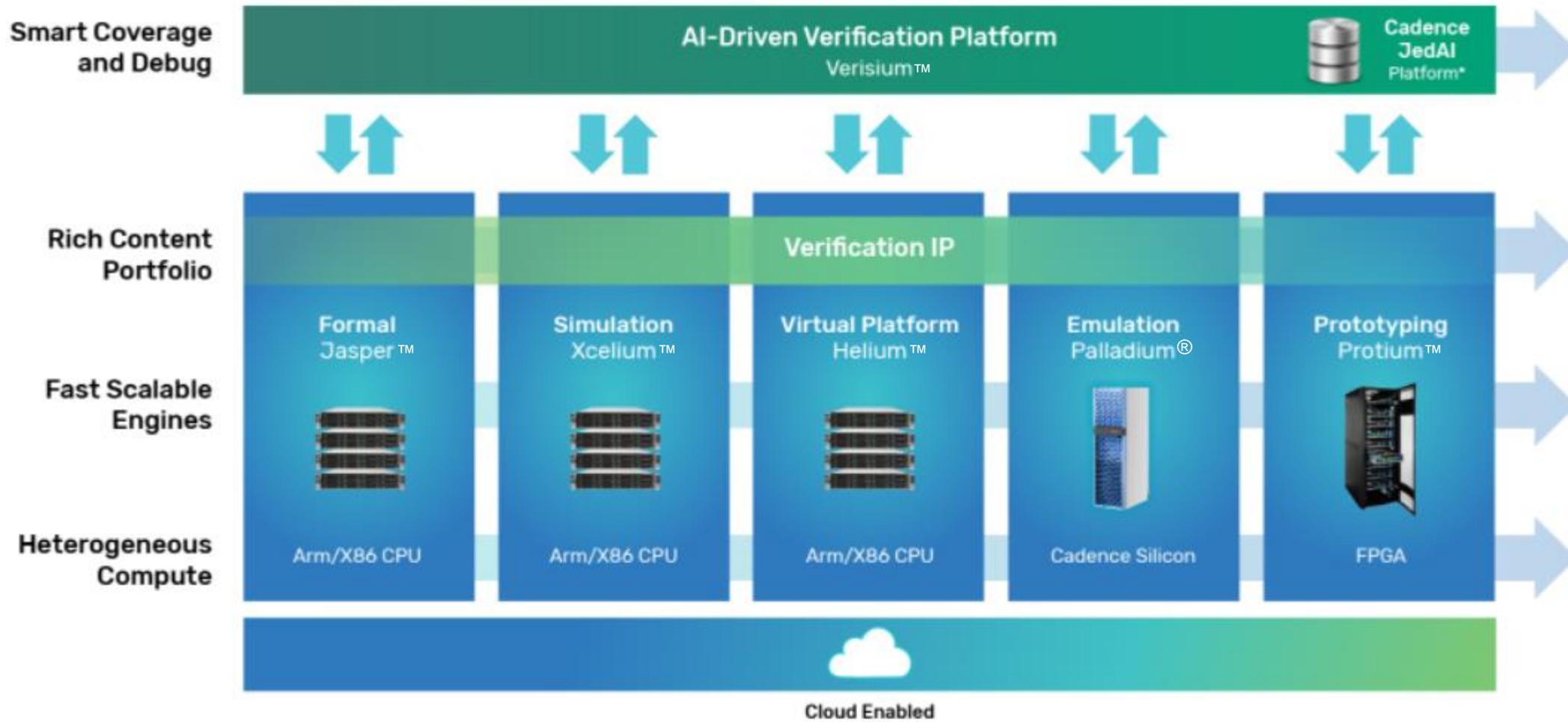
Supported by

cadence[®]

Agenda

- Verification Platform - Overview
- Functional Verification Methodologies
- Verification Environment
- Metrics Driven Verification
- Gate Level Simulation
- Debugging Methods
- Formal Verification
- Regression Environment

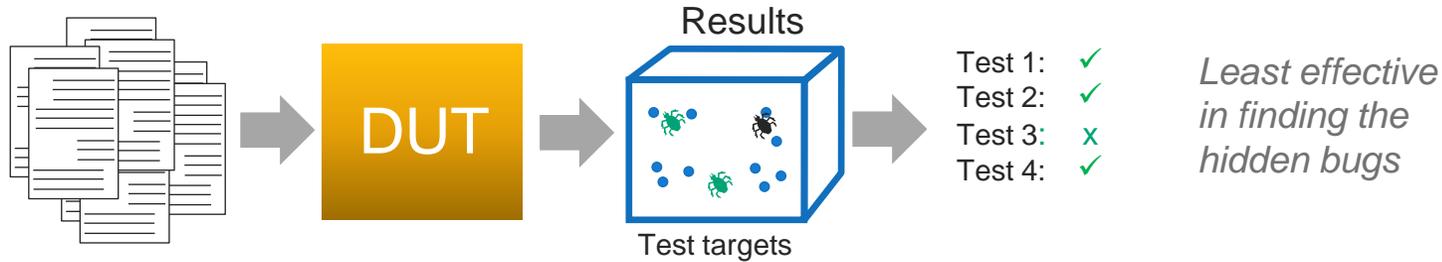
Verification Platform - Overview



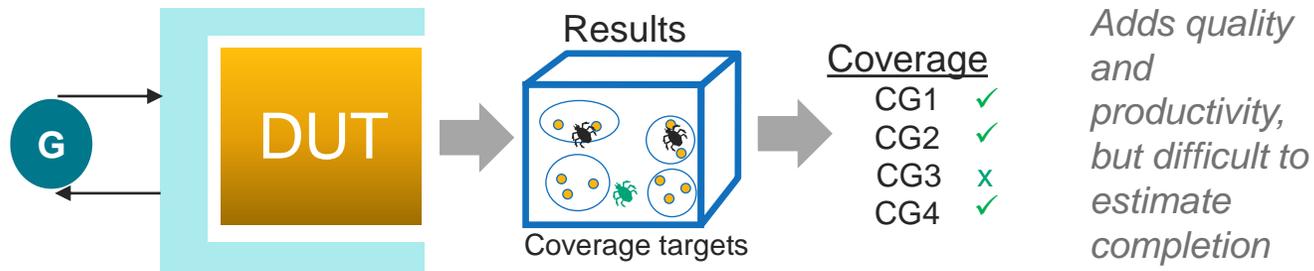
*Cadence Joint Enterprise Data and AI (JedAI) Platform

Functional Verification Methodologies

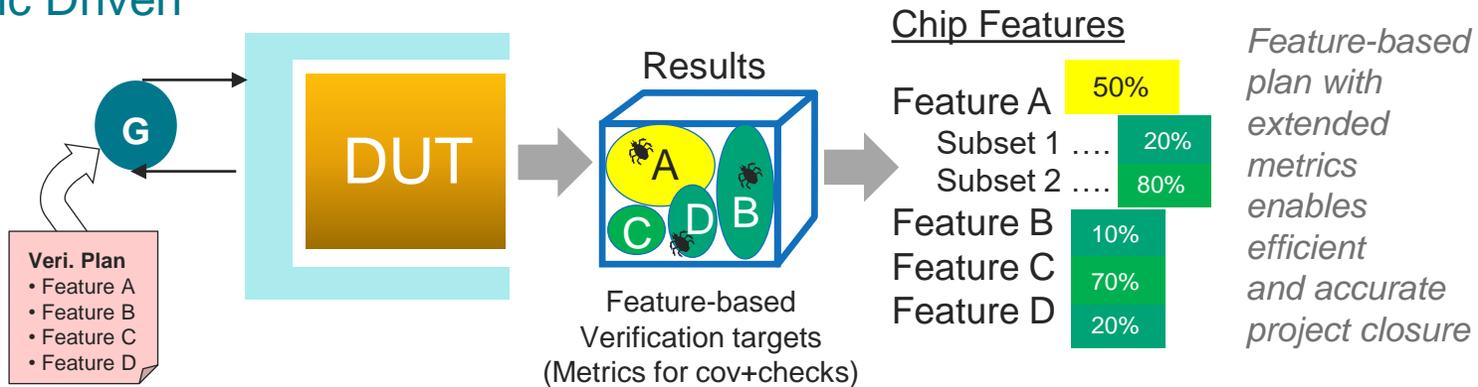
Directed Tests Driven



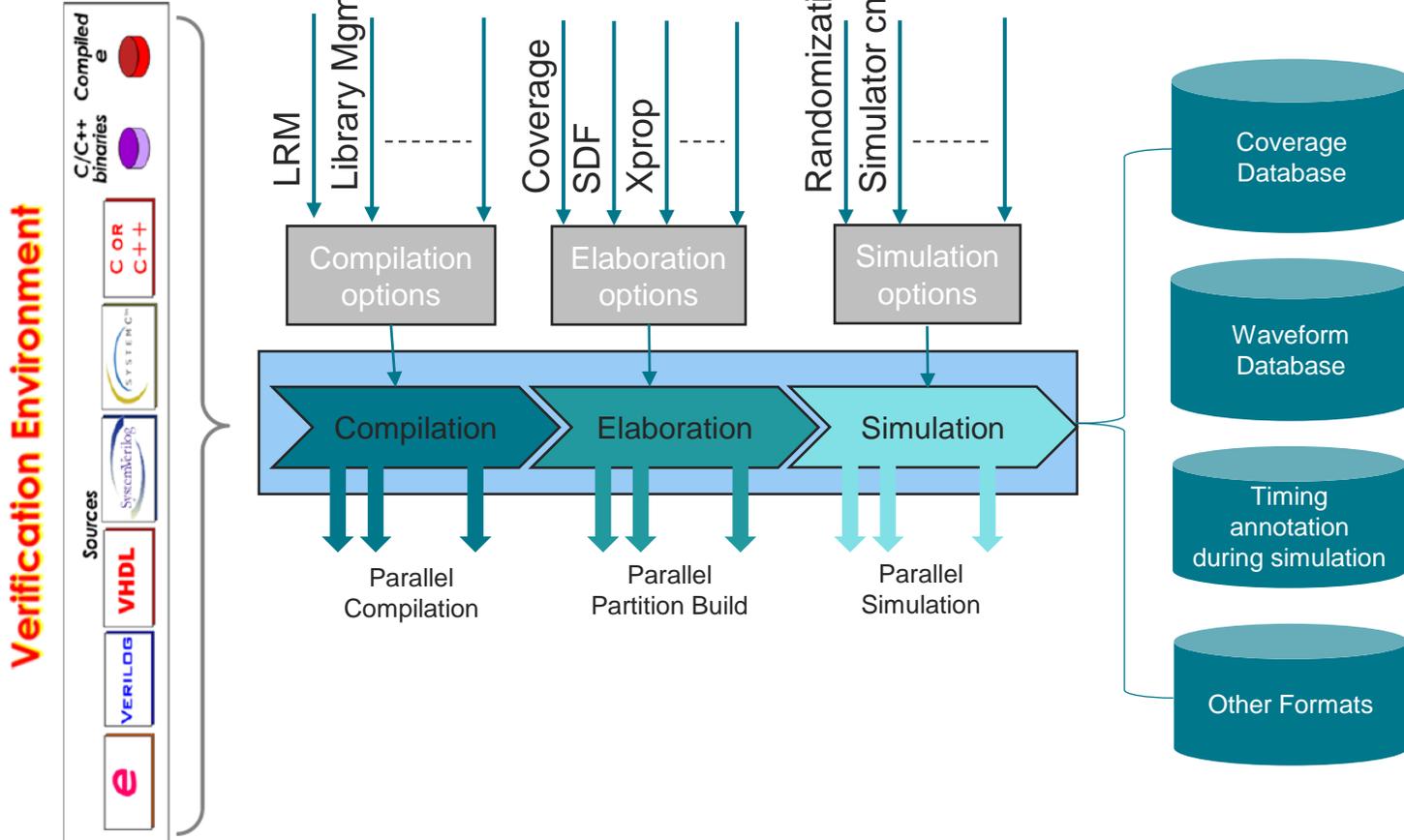
Coverage Driven



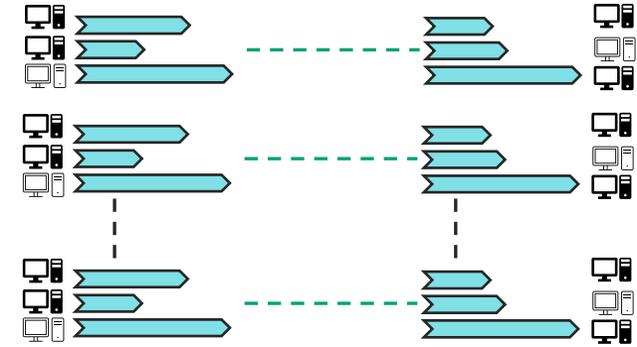
Metric Driven



Verification Environment

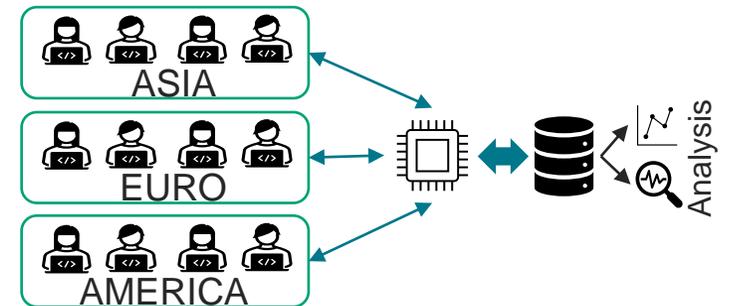


Execution Environment



DRM – Farms are used to execute such heavy computational tasks

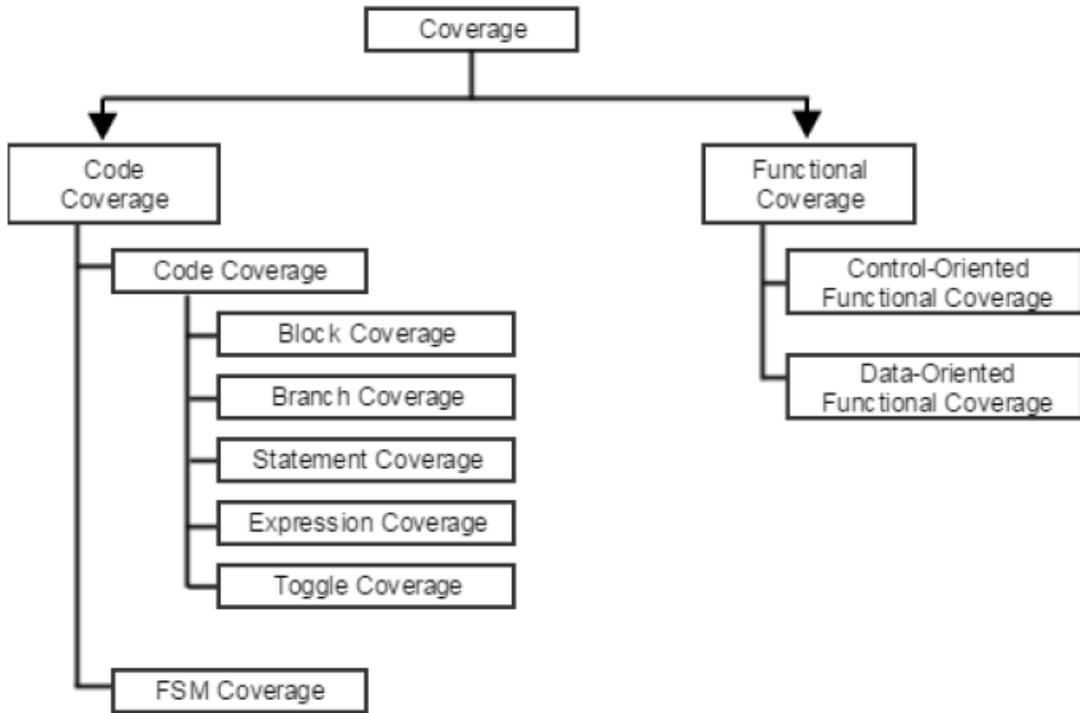
Global Connect



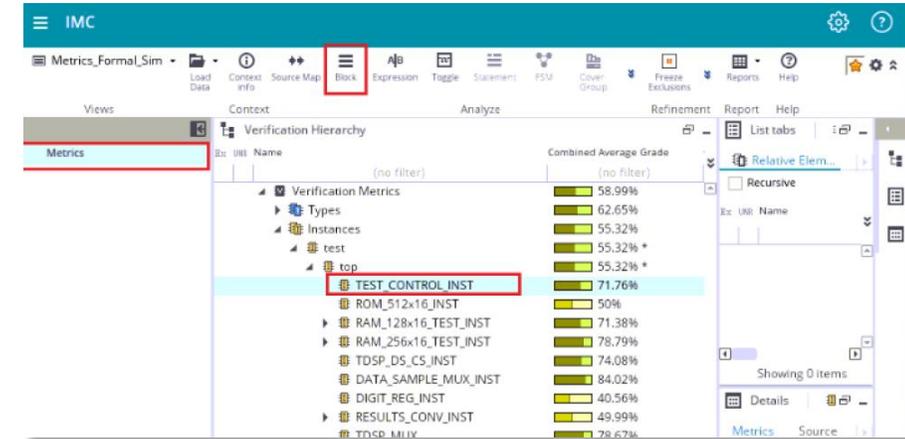
A powerful mechanism is needed to connect and collect data from global users.

Metrics-Driven Verification

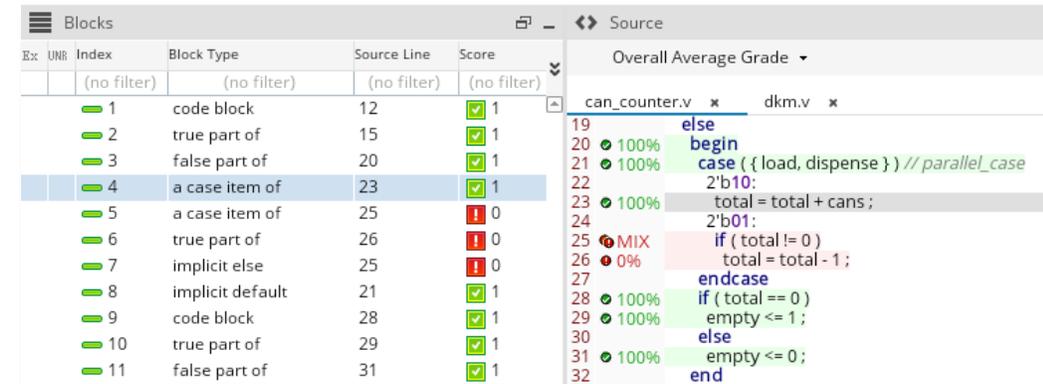
- Code coverage
 - This method assesses how well the test cases test the intended behavior
 - The extent to which they execute the design



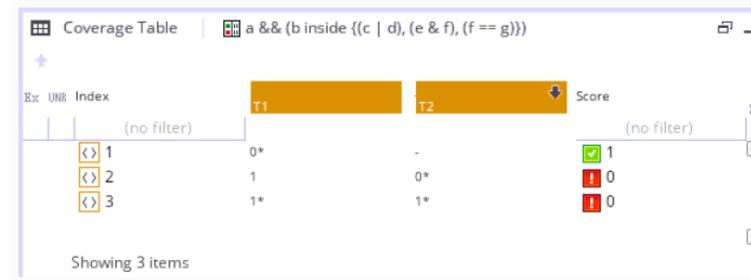
Coverage Hierarchy



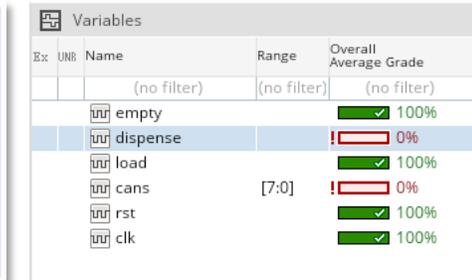
Block Coverage



Expression Coverage

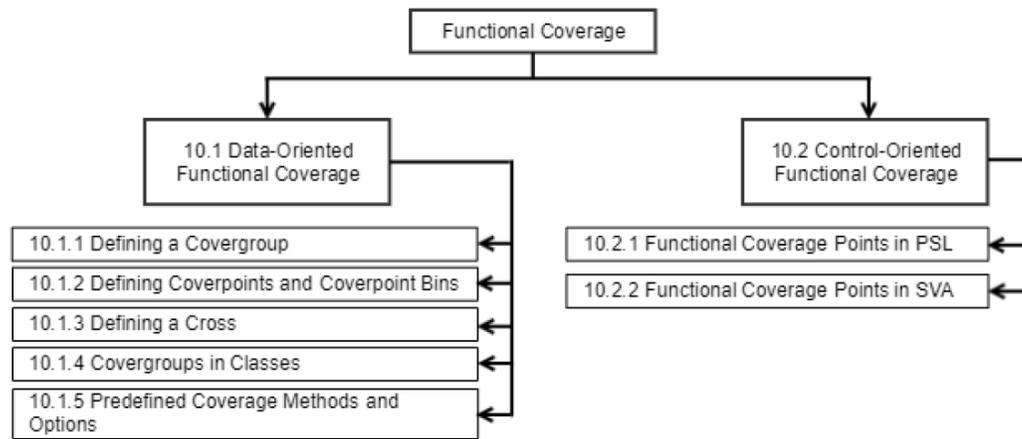


Toggle Coverage



Metrics-Driven Verification

- Functional coverage
 - Focuses on the functional aspects of a design
 - Helps in verification goals set by a test plan



Covergroup Coverage

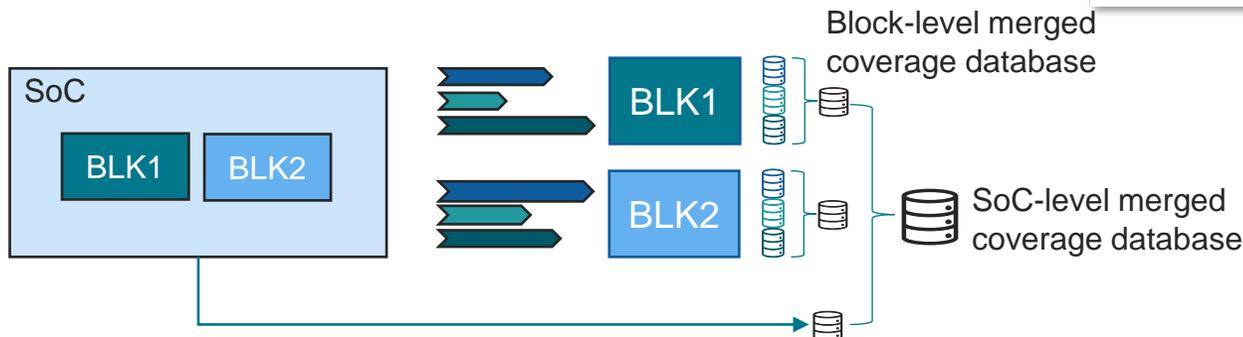
Ex: UIN	Overall Covered	Overall Average Grade	Name	Enclosing Entity
(no filter)	(no filter)	(no filter)	(no filter)	(no filter)
18 / 64 (28.12%)	28.12%	28.12%	reg_model_uart_ctrl_rf.ua_div_latch0.wcov	uvm_pkg
29 / 64 (45.31%)	45.31%	45.31%	reg_model_uart_ctrl_rf.ua_div_latch0.rcov	uvm_pkg
1 / 64 (1.56%)	1.56%	1.56%	reg_model_uart_ctrl_rf.ua_div_latch1.wcov	uvm_pkg
0 / 64 (0%)	0%	0%	reg_model_uart_ctrl_rf.ua_div_latch1.rcov	uvm_pkg
0 / 8 (0%)	0%	0%	reg_model_uart_ctrl_rf.ua_int_id.rcov	uvm_pkg
0 / 8 (0%)	0%	0%	reg_model_uart_ctrl_rf.ua_fifo_rcov	uvm_pkg
8 / 16 (50%)	53.57%	53.57%	reg_model_uart_ctrl_rf.ua_lcr.wcov	uvm_pkg

Ex: UIN	Name	Overall Average Grade	Overall Covered
(no filter)	(no filter)	(no filter)	(no filter)
div_val	div_val	45.31%	29 / 64 (45.31%)


```

assert.sv
5
6   always #1 clk = -clk;
7
8   always @(posedge clk)
9   begin
10    if (grant == 1) begin
11     CHECK_REQ_WHEN_GNT : assert (grant && request) begin
12      current_time = $time;
13      $display ("Seems to be working as expected at time %0t",current_time);
14    end else begin
15     current_time = $time;
16     #1 $error("assert failed at time %0t", current_time);
17    end
18  end
19  end
20
    
```

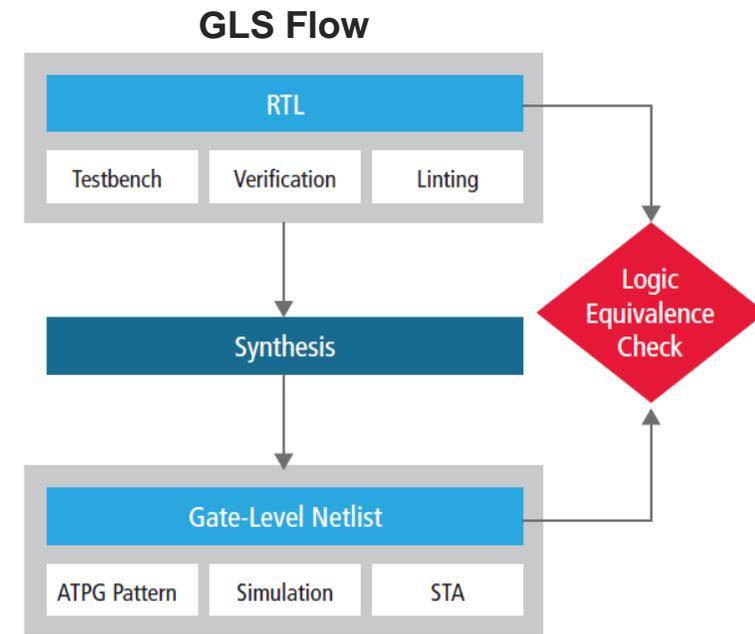
Merging of Coverage Database



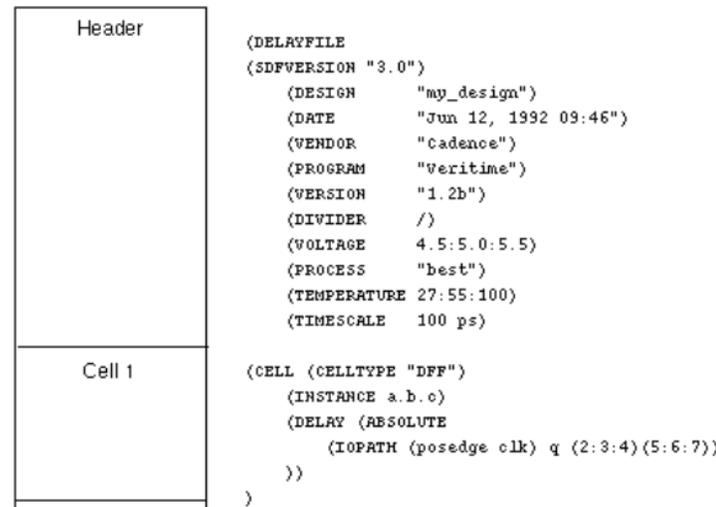
Assertion Coverage

Gate-Level Simulation

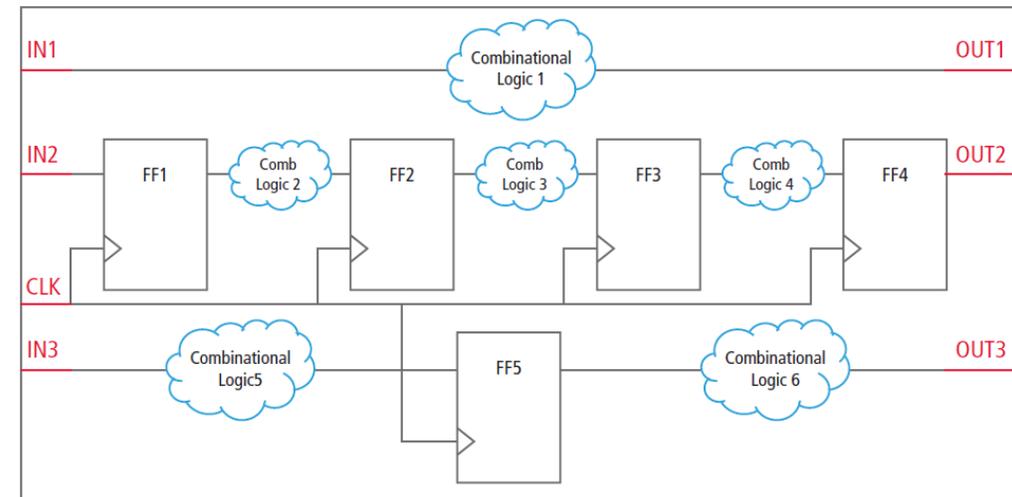
- Simulation of a netlist, with or without gate delays
- To ensure that the desired functionality is not lost in the translation from high-level RTL to low-level gate implementation
- GLS is done after the synthesis of the RTL code or post-P&R
- Timing Annotation
 - Specific Block
 - SDF (Standard Delay Format)
- SDF Content
 - IO PATH
 - TIMING CHECK



SDF Sample

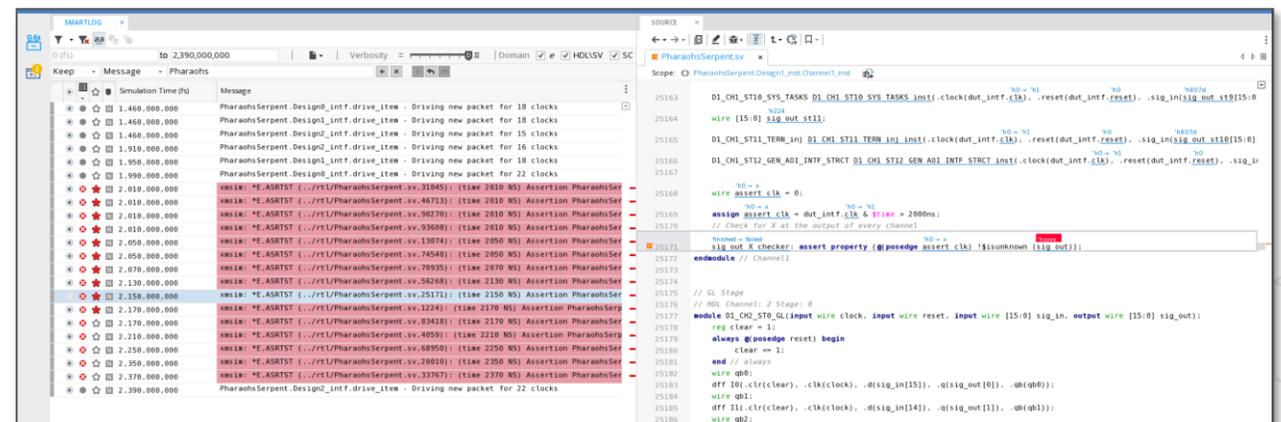
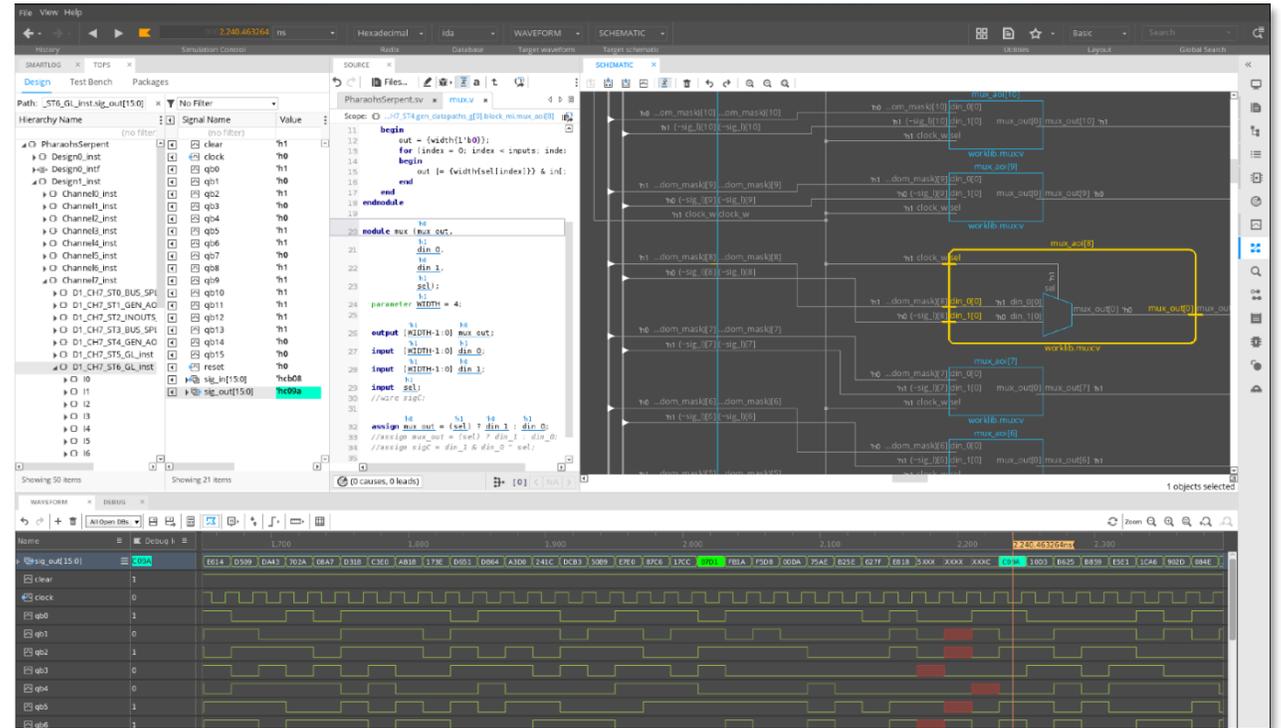


Design with sequential and combo blocks



Debugging Methods

- Code traversing
 - Code debug at each simulation time
- Waveform Debug
 - Analyzing dependent signals
 - Transaction, Mnemonic
- Schematic Tracing
 - Drivers and Loads
 - Connectivity verification (GLS)
- Analyzing Simulation log
 - Without running the simulation
 - Time traversing



Debugging Methods

High-Performance GUI (Launching, Analysis/Search)

Quickly launch supporting debug tools

Hyperlinked Log File

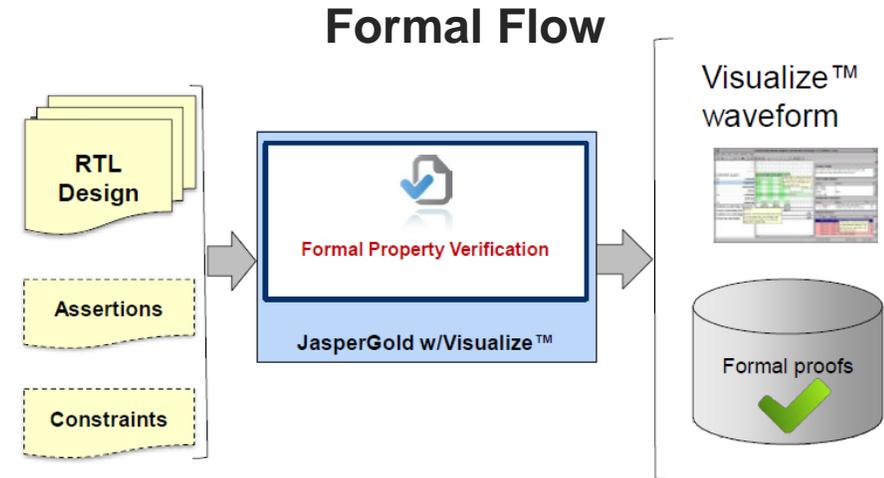
Source Browser

Navigation of Design/TB Hierarchy

Waveform Window

Formal Verification

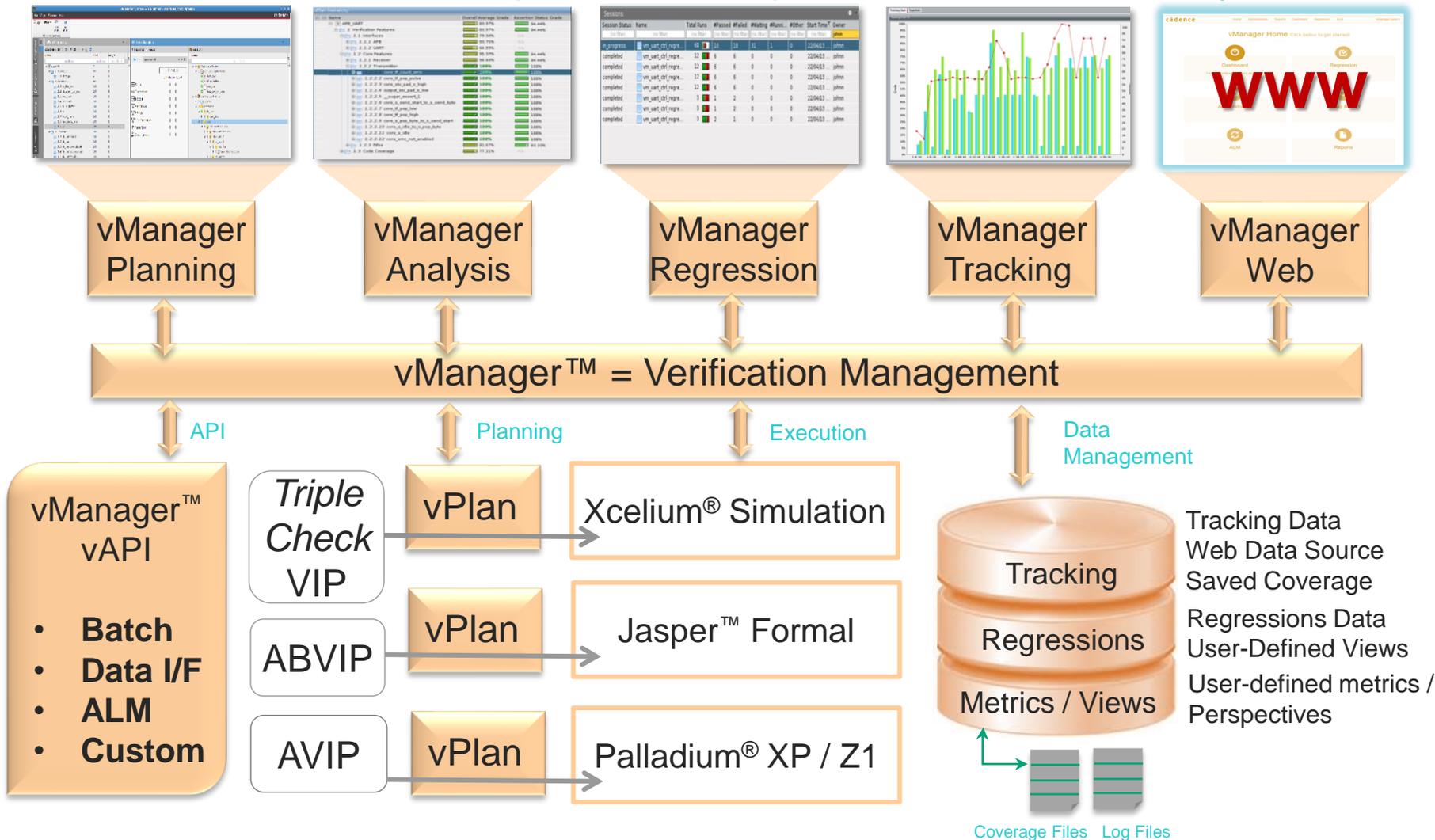
- Checks exhaustive if a **model** meets a given **spec**
 - Model: Synthesizable RTL
 - Spec: Properties (Assertions and Covers)
- Formal Analysis
 - Formal tests all possible stimuli, one cycle at a time
- Properties
 - Formal checks properties as it is walking through stimuli
- Constraints
 - Not all input stimulus combinations are legal
 - Assume properties tell formal what is legal



 <p>Formal Property Verification App</p> <ul style="list-style-type: none"> • Block-level or end-to-end properties • Interactive debug, what-if & constraint setting • High performance and capacity 	 <p>Superlint (AFL) App</p> <ul style="list-style-type: none"> • Automatic RTL checks • Overflow, dead code, livelock/deadlock... • Integrated waiver handling and debug 	 <p>X-Propagation Verification App</p> <ul style="list-style-type: none"> • Automatic property generation • Unexpected X detection & debug 	 <p>CSR Control/Status Register Verif. App</p> <ul style="list-style-type: none"> • IP-XACT input • Comprehensive access policy checks • Standard & proprietary protocols 	 <p>Connectivity Verification App</p> <ul style="list-style-type: none"> • Excel or IP-XACT input • Sub-system and chip-level connectivity • Conditional, combinational or sequential connections 	 <p>Coverage Unreachability App</p> <ul style="list-style-type: none"> • Proves reachability of coverage holes • Simulation coverage DB and RTL inputs • No formal expertise required
 <p>Design Coverage App</p> <ul style="list-style-type: none"> • Provides formal coverage metrics • Analyzes property set completeness • Shows verification from bounded proofs 	 <p>Sequential Equivalence Checking App</p> <ul style="list-style-type: none"> • Sequential, temporal & functional equivalence • Reference versus modified RTL • Side-by-side debug • Full chip capacity 	 <p>Clock Domain Crossing App</p> <ul style="list-style-type: none"> • Full structural and functional checks • Metastability injection flow with Xcelium™ • State-of-the-art graphical debug & waiver handling 	 <p>Functional Safety Verification App</p> <ul style="list-style-type: none"> • Formally analyzes fault categorization • Complementary to Xcelium Safety • Graphical debug & full reporting 	 <p>Low Power Verification App</p> <ul style="list-style-type: none"> • Verifies power-aware formal model • RTL and power intent file inputs • Structural, functional, & sequence checks 	 <p>Security Path Verification App</p> <ul style="list-style-type: none"> • Identifies secure data leakage paths • Verifies data sanctity & fault-tolerant security

Regression Environment

Create plans, Analyze metrics, Launch jobs, view Results, triage data Track progress, Submit reports Dashboards Reports, Launch





cādence[®]

© 2023 Cadence Design Systems, Inc. All rights reserved worldwide. Cadence, the Cadence logo, and the other Cadence marks found at <https://www.cadence.com/go/trademarks> are trademarks or registered trademarks of Cadence Design Systems, Inc. Accellera and SystemC are trademarks of Accellera Systems Initiative Inc. All Arm products are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All MIPI specifications are registered trademarks or service marks owned by MIPI Alliance. All PCI-SIG specifications are registered trademarks or trademarks of PCI-SIG. All other trademarks are the property of their respective owners.