



Digital Verification

Budi Murty
Aug 1, 2023

 **cā dence**[®]

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



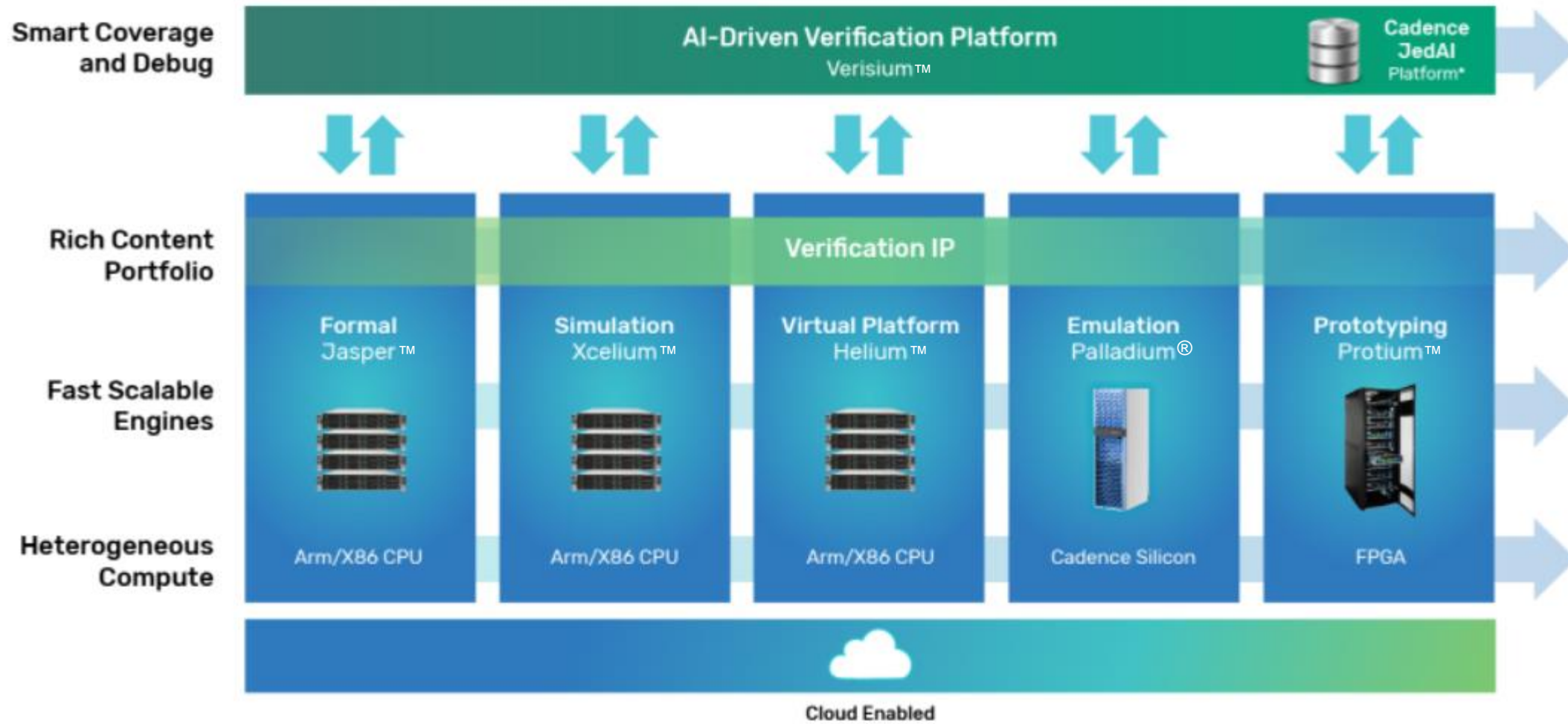
Supported by

cādence®

Agenda

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

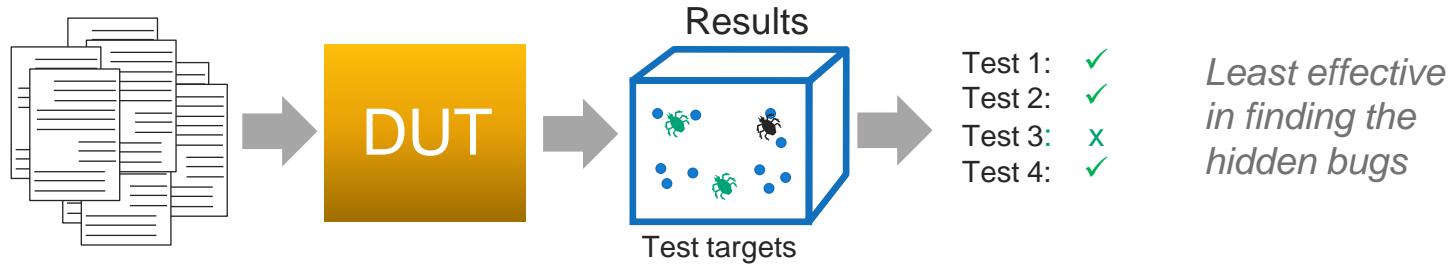
Verification Platform - Overview



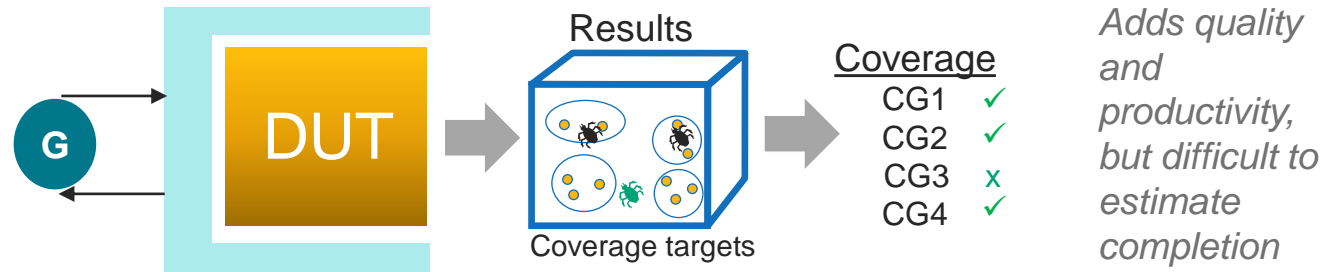
*Cadence Joint Enterprise Data and AI (JedAI) Platform

Functional Verification Methodologies

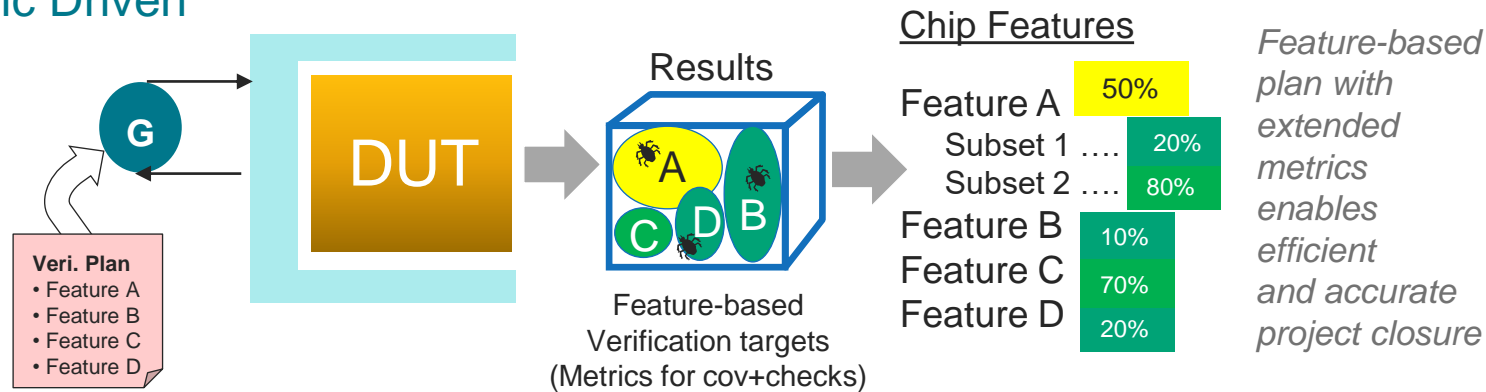
Directed Tests Driven



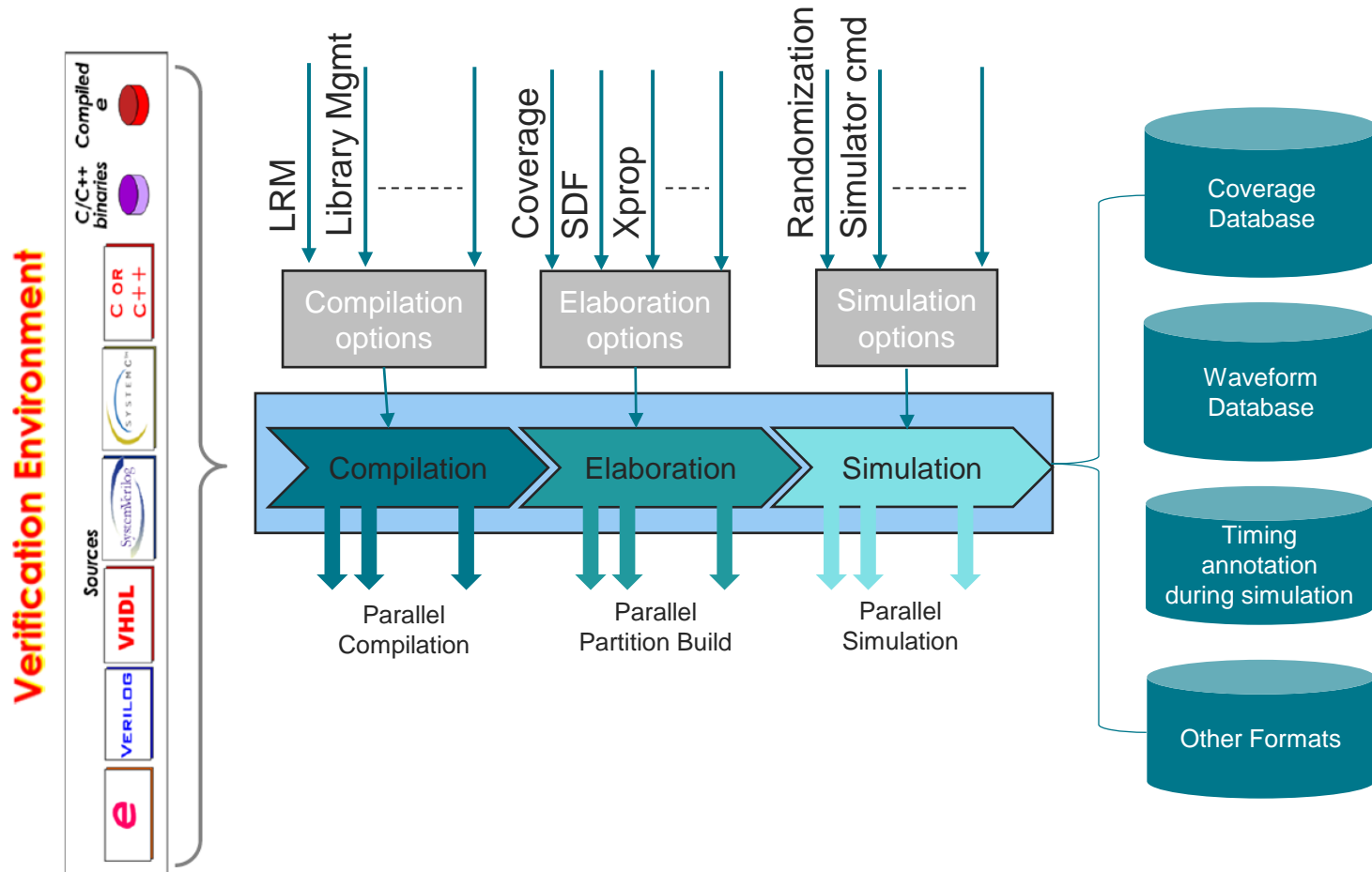
Coverage Driven



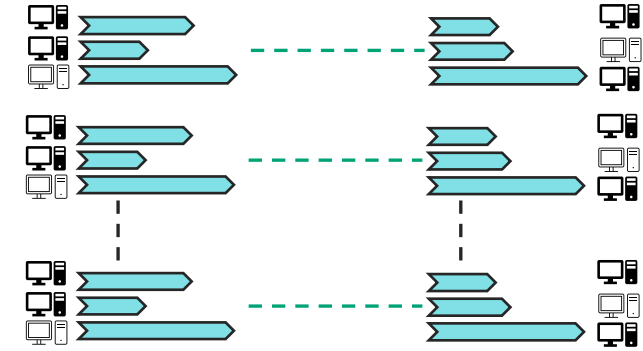
Metric Driven



Verification Environment

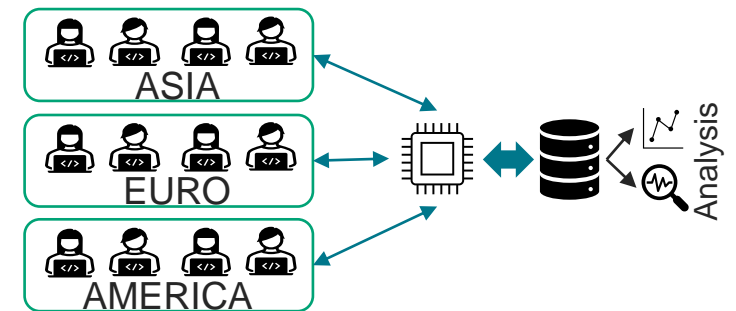


Execution Environment



DRM – Farms are used to execute such heavy computational tasks

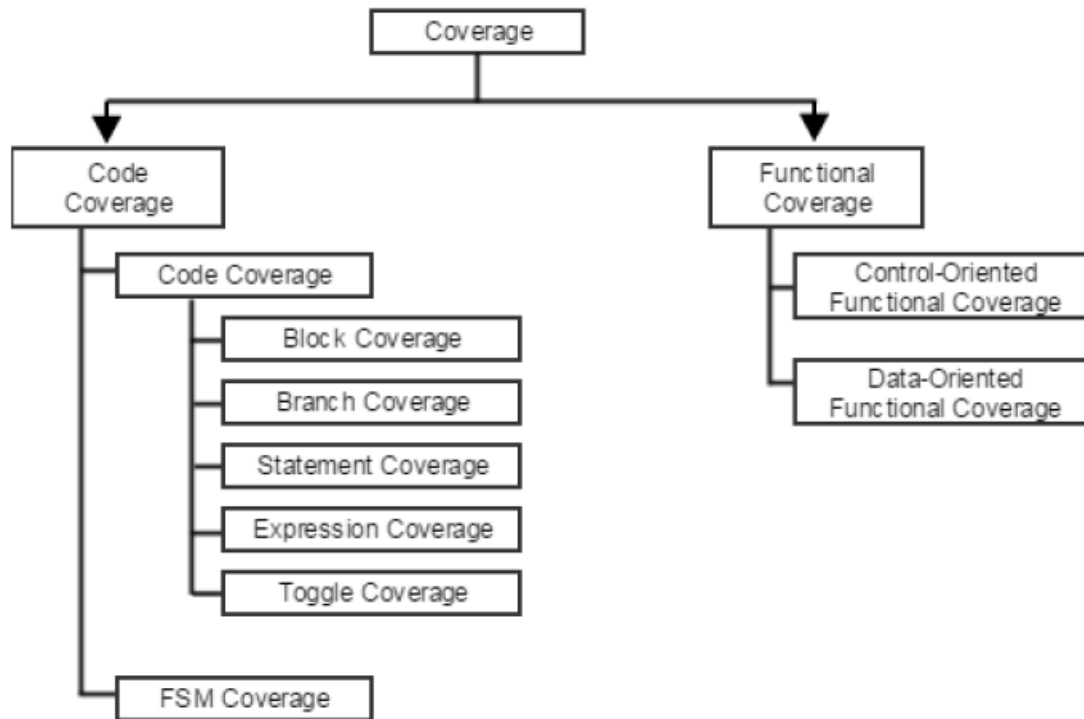
Global Connect



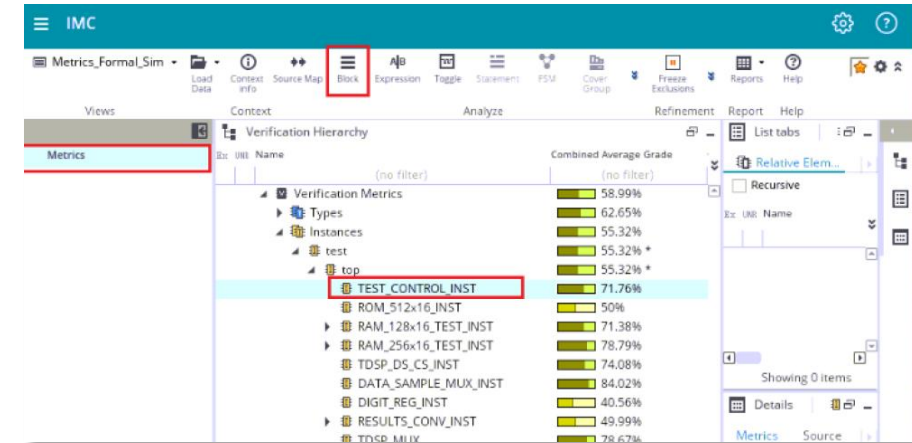
A powerful mechanism is needed to connect and collect data from global users.

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

Blocks					Source	
Bx	UNR	Index	Block Type	Source Line	Score	Overall Average Grade
		(no filter)	(no filter)	(no filter)	(no filter)	
		1	code block	12	✓ 1	<div>can_counter.v</div> <div>dkm.v</div> <pre> 19 else 20 begin 21 case ({ load, dispense }) // parallel_case 22 2'b10: 23 total = total + cans; 24 2'b01: 25 if (total != 0) 26 total = total - 1; 27 endcase 28 if (total == 0) 29 empty <= 1; 30 else 31 empty <= 0; 32 end </pre>
		2	true part of	15	✓ 1	
		3	false part of	20	✓ 1	
		4	a case item of	23	✓ 1	
		5	a case item of	25	! 0	
		6	true part of	26	! 0	
		7	implicit else	25	! 0	
		8	implicit default	21	✓ 1	
		9	code block	28	✓ 1	
		10	true part of	29	✓ 1	
		11	false part of	31	✓ 1	

Expression Coverage

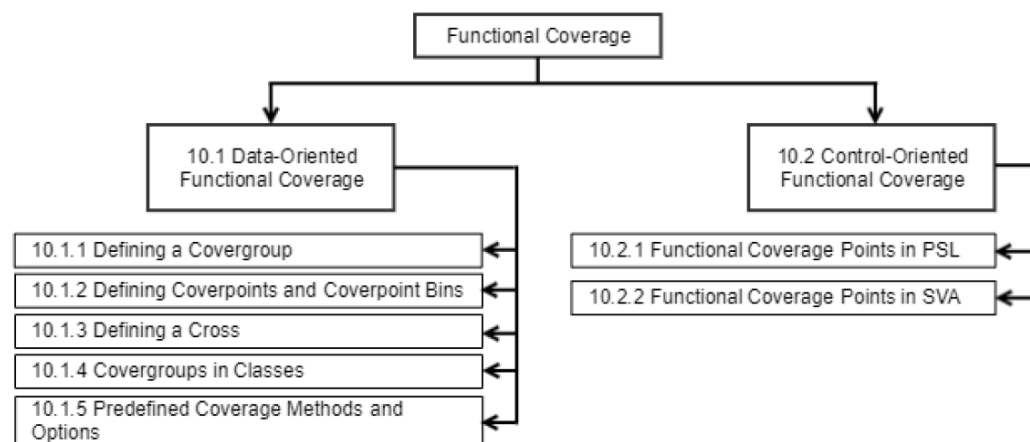
Coverage Table					a && (b inside {(c d), (e & f), (f == g)})	
Bx	UNR	Index	T1	T2	Score	
		(no filter)	(no filter)	(no filter)	(no filter)	
		1	0*	-	✓ 1	<div>Showing 3 items</div>
		2	1	0*	! 0	
		3	1*	1*	! 0	

Toggle Coverage

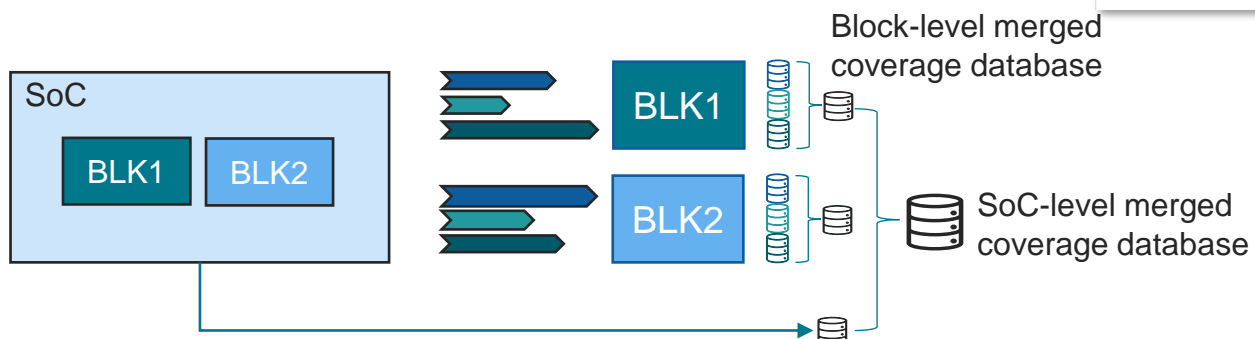
Variables					Overall Average Grade	
Bx	UNR	Name	Range		(no filter)	
		(no filter)	(no filter)	(no filter)	(no filter)	
		empty			✓ 100%	<div> <div>dispense</div> <div>load</div> <div>cans</div> <div>rst</div> <div>clk</div> </div>
		dispense			! 0%	
		load			✓ 100%	
		cans	[7:0]		! 0%	
		rst			✓ 100%	
		clk			✓ 100%	

Metrics-Driven Verification

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



Merging of Coverage Database



Covergroup Coverage

Cover Groups				Assertions			
Ex: UNB	Overall Covered	Overall Average Grade	Name	Enclosing Entity	Abstract	Expand	
(no filter)	(no filter)	(no filter)	(no filter)		Ex: UNB	Name	Overall Average Grade
18 / 64 (28.12%)	28.12%		reg_model_uart_ctrl_rf.ua_div_latch0.wcov	uvm_pkg	auto[0:3]		100%
29 / 64 (45.31%)	45.31%		reg_model_uart_ctrl_rf.ua_div_latch0.rcov	uvm_pkg	auto[4:7]		100%
1 / 64 (1.56%)	1.56%		reg_model_uart_ctrl_rf.ua_div_latch1.wcov	uvm_pkg	auto[8:11]		0%
0 / 64 (0%)	0%		reg_model_uart_ctrl_rf.ua_div_latch1.rcov	uvm_pkg	auto[12:15]		0%
0 / 8 (0%)	0%		reg_model_uart_ctrl_rf.ua_int_id.rcov	uvm_pkg	auto[16:19]		100%
0 / 8 (0%)	0%		reg_model_uart_ctrl_rf.ua_fifo_ctrl.wcov	uvm_pkg	auto[20:23]		100%
8 / 16 (50%)	53.57%		reg_model_uart_ctrl_rf.ua_lcr.wcov	uvm_pkg			

Items	reg_model_uart_ctrl_rf.ua_div_latch0.rcov		
Ex: UNB	Name	Overall Average Grade	Overall Covered
(no filter)	(no filter)	(no filter)	(no filter)
div_val		45.31%	29 / 64 (45.31%)

(no filter)	(no filter)	
Verification Metrics	89.17%	
Types	89.17%	
Instances	89.17%	
assert_immediate	89.17%	
CHECK_REQ_WHEN_GNT	100%	


```

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
    
```

Assertion Coverage

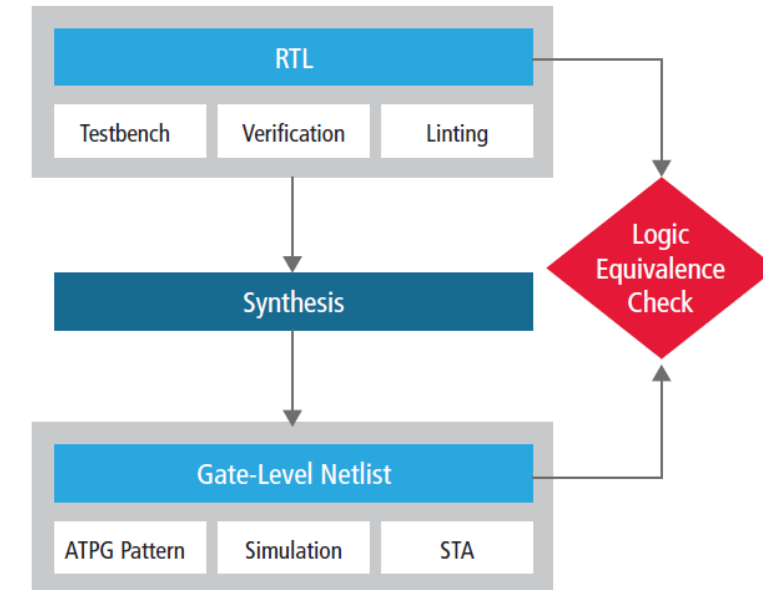
Gate-Level Simulation

- 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

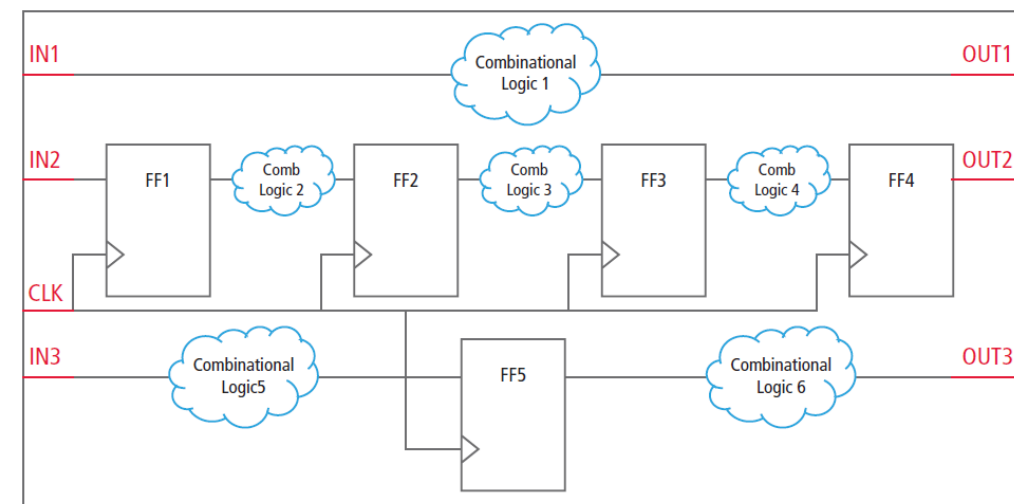
SDF Sample

Header	<pre><DELAYFILE (SDFVERSION "3.0") (DSIGN "my_design") (DATE "Jun 12, 1992 09:46") (VENDOR "Cadence") (PROGRAM "Veritime") (VERSION "1.2b") (DIVIDER /) (VOLTAGE 4.5:5.0:5.5) (PROCESS "best") (TEMPERATURE 27:55:100) (TIMESCALE 100 ps)</pre>
Cell 1	<pre><CELL (CELLTYPE "DFF") (INSTANCE a.b.c) (DELAY (RESOLUTE (IOPATH (posedge clk) q {2:3:4}{5:6:7})))))</pre>

GLS Flow

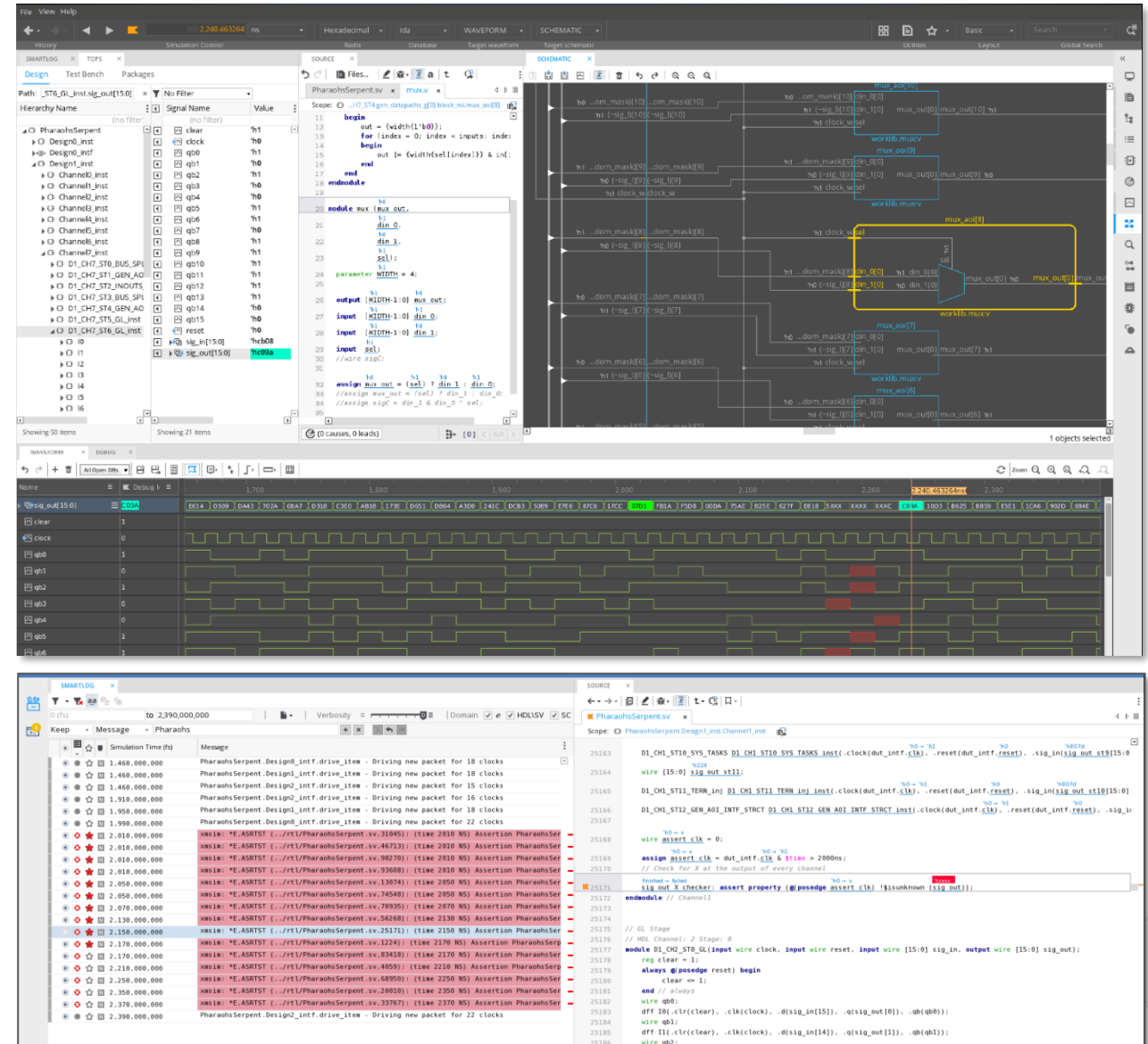


Design with sequential and combo blocks



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

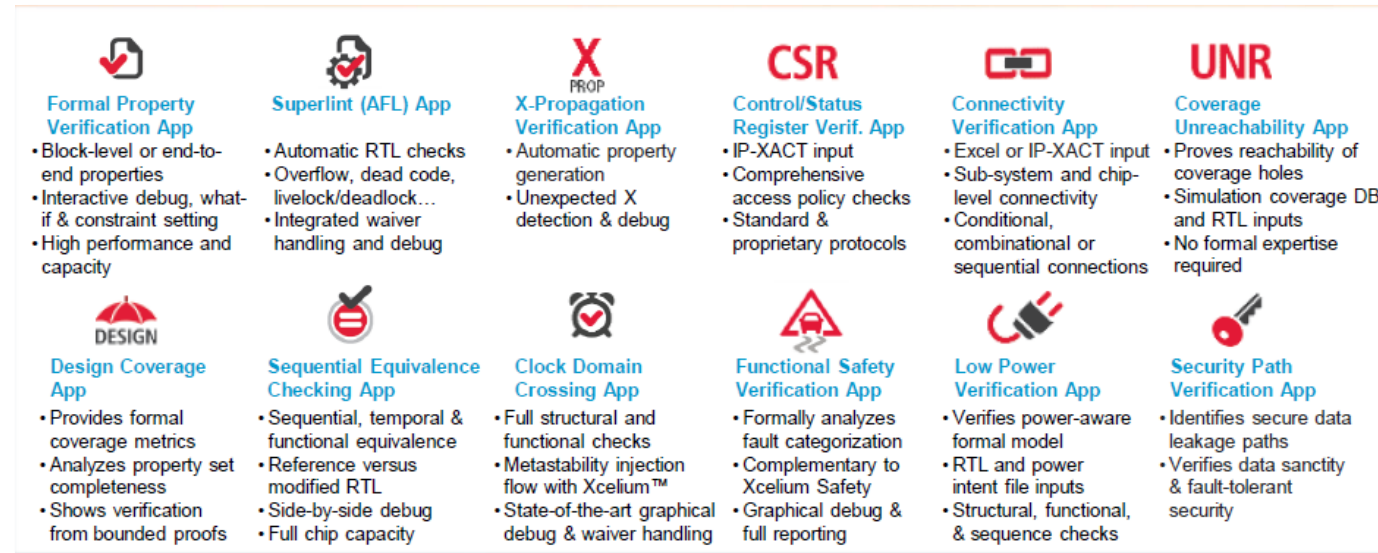
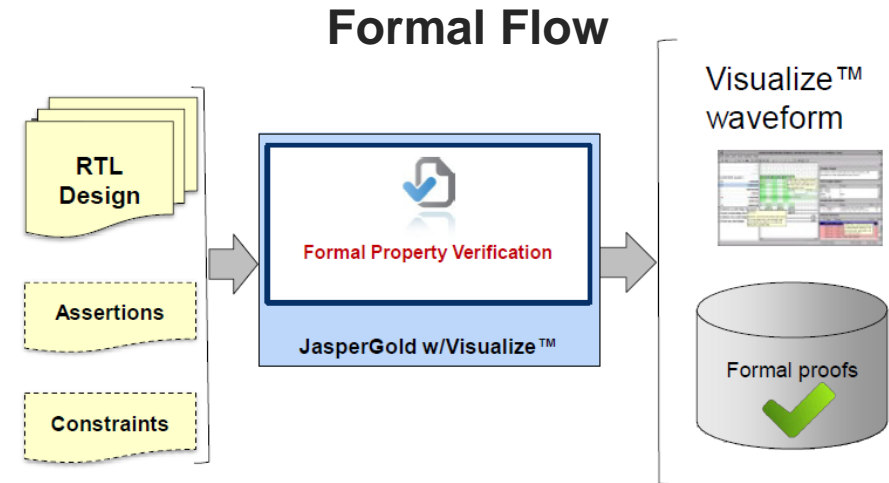
Source Browser

Navigation of
Design/TB Hierarchy

Waveform Window

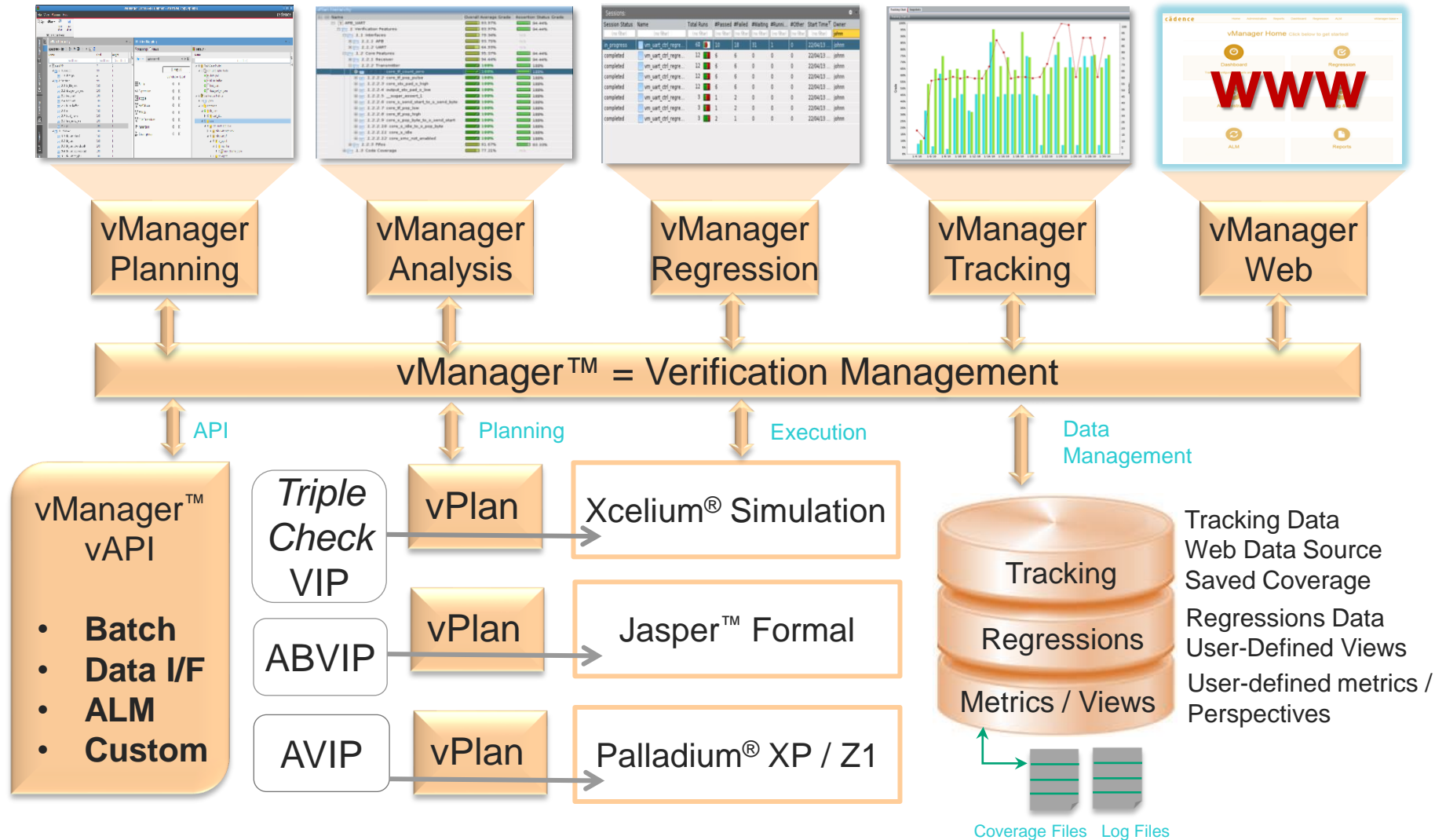
Formal Verification

- 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



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.

