Next-Generation Verdi: Overview of the IDE and the Verification Management System

Jens Dickel, Ionut Cirjan, Noam Roth
Synopsys

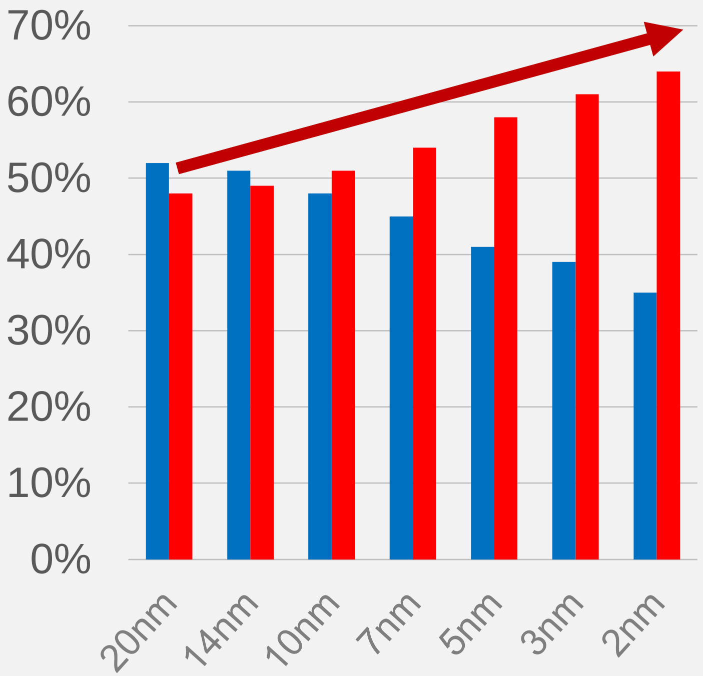
Agenda

- Introduction to Next-Generation Verdi
- Synopsys Verdi® Verification Management System with VC Execution Manger
- Verdi Integrated Design Environment (IDE) with Euclide
- Summary + Q&A

Introduction to Next-Generation Verdi

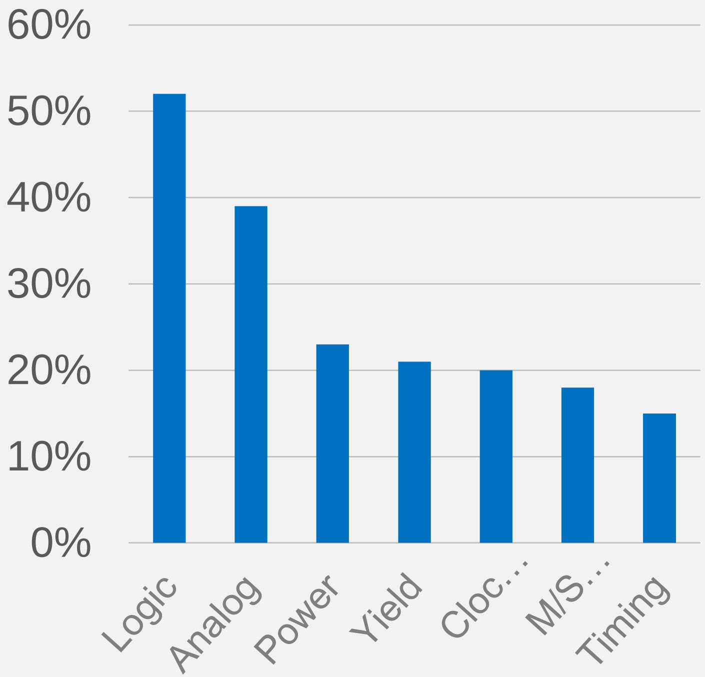
Impact on Right First-Time Silicon

Fewer First-Time Right



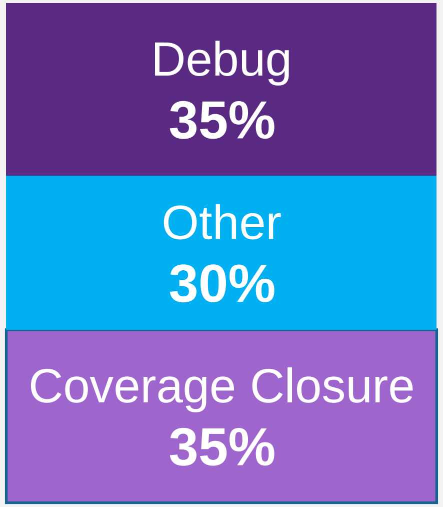
More Respins

Logic Bugs Dominating



Main Cause of Respins

Verification Time Spent

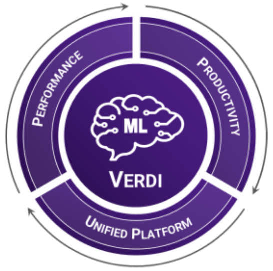


Need to Reduce Debug Time!

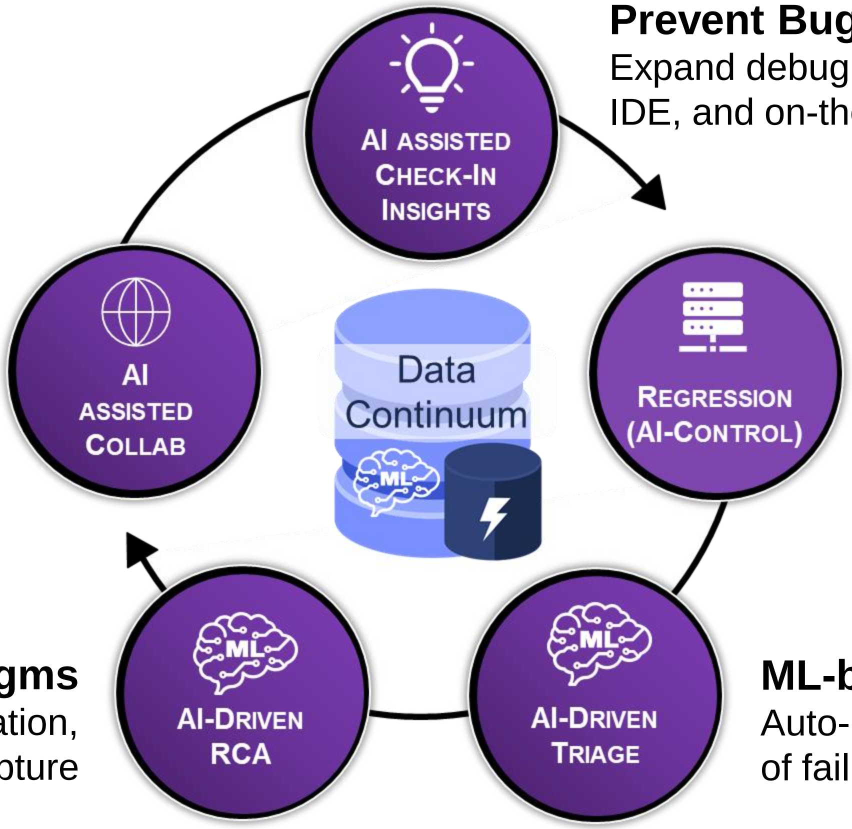
Source: Wilson Report 2022

AI-Assisted Debug Flow

Next-Generation Debug: Improves debug productivity up to 10X



Prevent Bugs
Expand debug with RTL & TB IDE, and on-the-fly linting



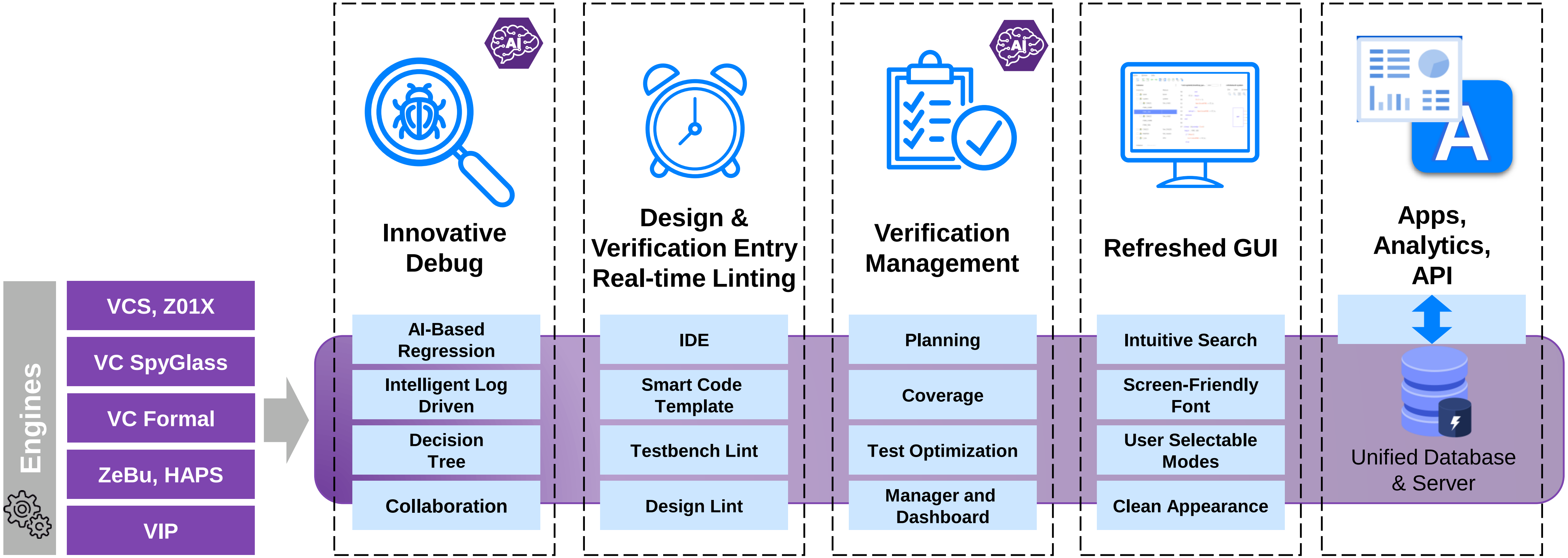
Global Collaboration
Multi-site, multi-user collaborative debug, knowledge reuse

Regression Management
ML-optimized daily regression, reduce HW costs and increase efficiency

New Debug Paradigms
Auto RCA, Smart Log visualization, debug knowledge capture

ML-based Triage
Auto-binning and triage, predict source of failures (RTL or engineer)

Introducing Next-Generation Verdi Platform



Engines

- VCS, Z01X
- VC SpyGlass
- VC Formal
- ZeBu, HAPS
- VIP

Innovative Debug

- AI-Based Regression
- Intelligent Log Driven
- Decision Tree
- Collaboration

Design & Verification Entry Real-time Linting

- IDE
- Smart Code Template
- Testbench Lint
- Design Lint

Verification Management

- Planning
- Coverage
- Test Optimization
- Manager and Dashboard

Refreshed GUI

- Intuitive Search
- Screen-Friendly Font
- User Selectable Modes
- Clean Appearance

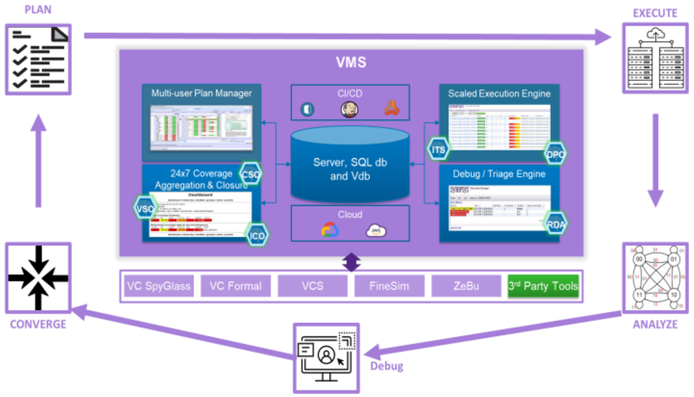
Apps, Analytics, API

Unified Database & Server

Verdi Verification Management System

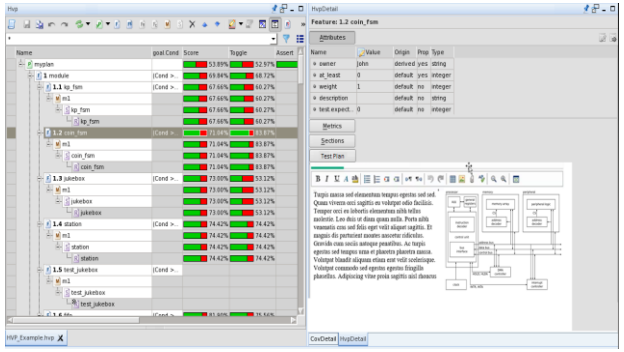
Verification Management System

Manager and Dashboard



- Test planning, execution & debug, coverage merge and annotation
- Enables verification data-over-time to be mined for analytics

Planner



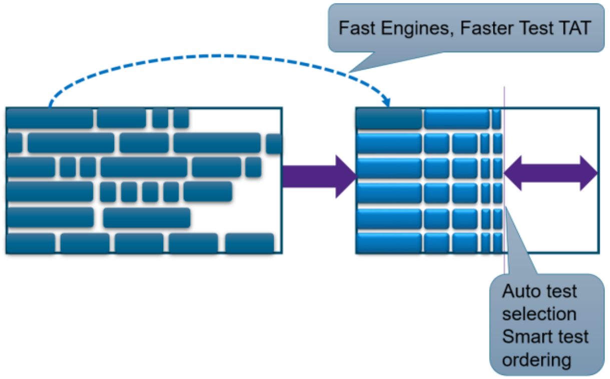
Name	goal.Cond	Score	Toggle	Assert
myplan		53.89%	52.97%	
1 module	(Cond >...	69.84%	68.72%	
1.1 kp_fsm	(Cond >...	67.66%	60.27%	
m1		67.66%	60.27%	
kp_fsm		67.66%	60.27%	
kp_fsm		67.66%	60.27%	
1.2 coin_fsm	(Cond >...	71.04%	83.87%	
m1		71.04%	83.87%	
coin_fsm		71.04%	83.87%	
coin_fsm		71.04%	83.87%	
1.3 jukebox	(Cond >...	73.00%	53.12%	
m1		73.00%	53.12%	
jukebox		73.00%	53.12%	
jukebox		73.00%	53.12%	
1.4 station	(Cond >...	74.42%	74.42%	
m1		74.42%	74.42%	
station		74.42%	74.42%	
station		74.42%	74.42%	
1.5 test_jukebox	(Cond >...			
m1				
test_jukebox				
test_jukebox				
1.6 fifo	(Cond >...	81.90%	75.56%	

Name	Value	Origin	Prop	Type
owner	john	derived	yes	string
at_least	0	default	yes	integer
weight	1	default	no	integer
description		default	no	string
test expect...	0	default	no	integer

Turpis massa sed elementum tempus egestas sed sed. Quam viverra orci sagittis eu volutpat odio facilisis. Tempor orci eu lobortis elementum nibh tellus molestie. Leo duis ut diam quam nulla. Porta nibh venenatis cras sed felis eget velit aliquet sagittis. Et magnis dis parturient montes nascetur ridiculus. Gravida cum sociis natoque penatibus. Ac turpis egestas sed tempus urna et pharetra pharetra massa. Volutpat blandit aliquam etiam erat velit scelerisque. Volutpat commodo sed egestas egestas fringilla phasellus. Adipiscing vitae proin sagittis nisl rhoncus

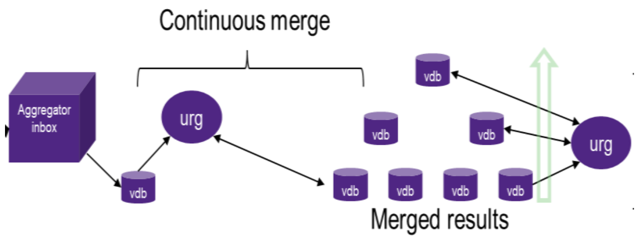
- Multi-user test scheduling/planning
- Supports change history and restore
- API for automated report generation and updates

Runner


Fast Engines, Faster Test TAT
Auto test selection Smart test ordering

- Runs regressions
- Order tests to eliminate long tail
- Synopsys VCS® engine performance enhancement

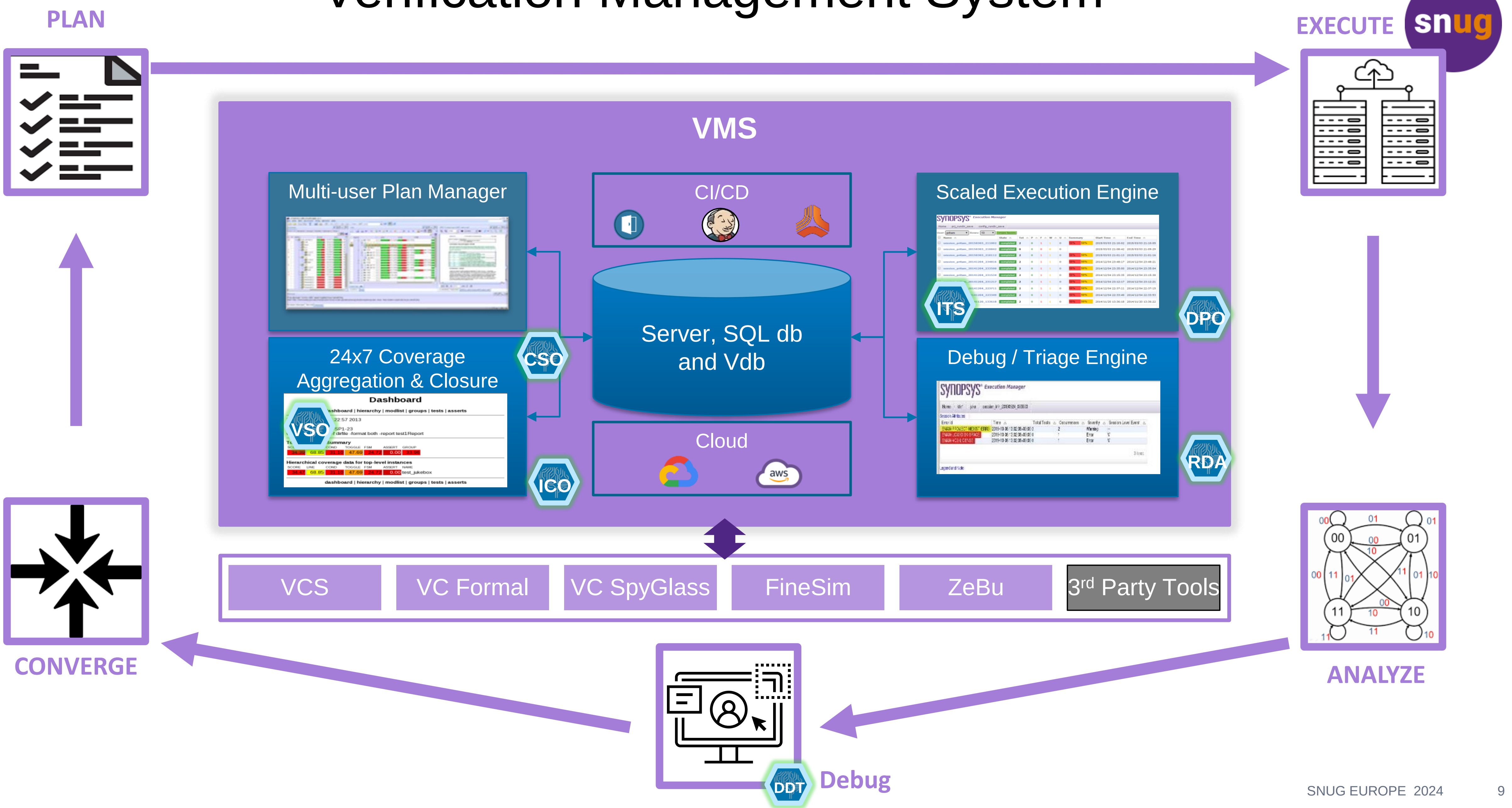
Coverage



24/7 Merge

- Continuously merges incoming coverage
- Integrated tagged VDB from ad hoc regression runs
- Can generate moving window merge VDB

Verification Management System



PLAN

EXECUTE

VMS

Multi-user Plan Manager

CI/CD

Scaled Execution Engine

Server, SQL db and Vdb

24x7 Coverage Aggregation & Closure

Debug / Triage Engine

Cloud

CSO

VSO

ICO

ITS

DPO

RDA

VCS

VC Formal

VC SpyGlass

FineSim

ZeBu

3rd Party Tools

CONVERGE

ANALYZE

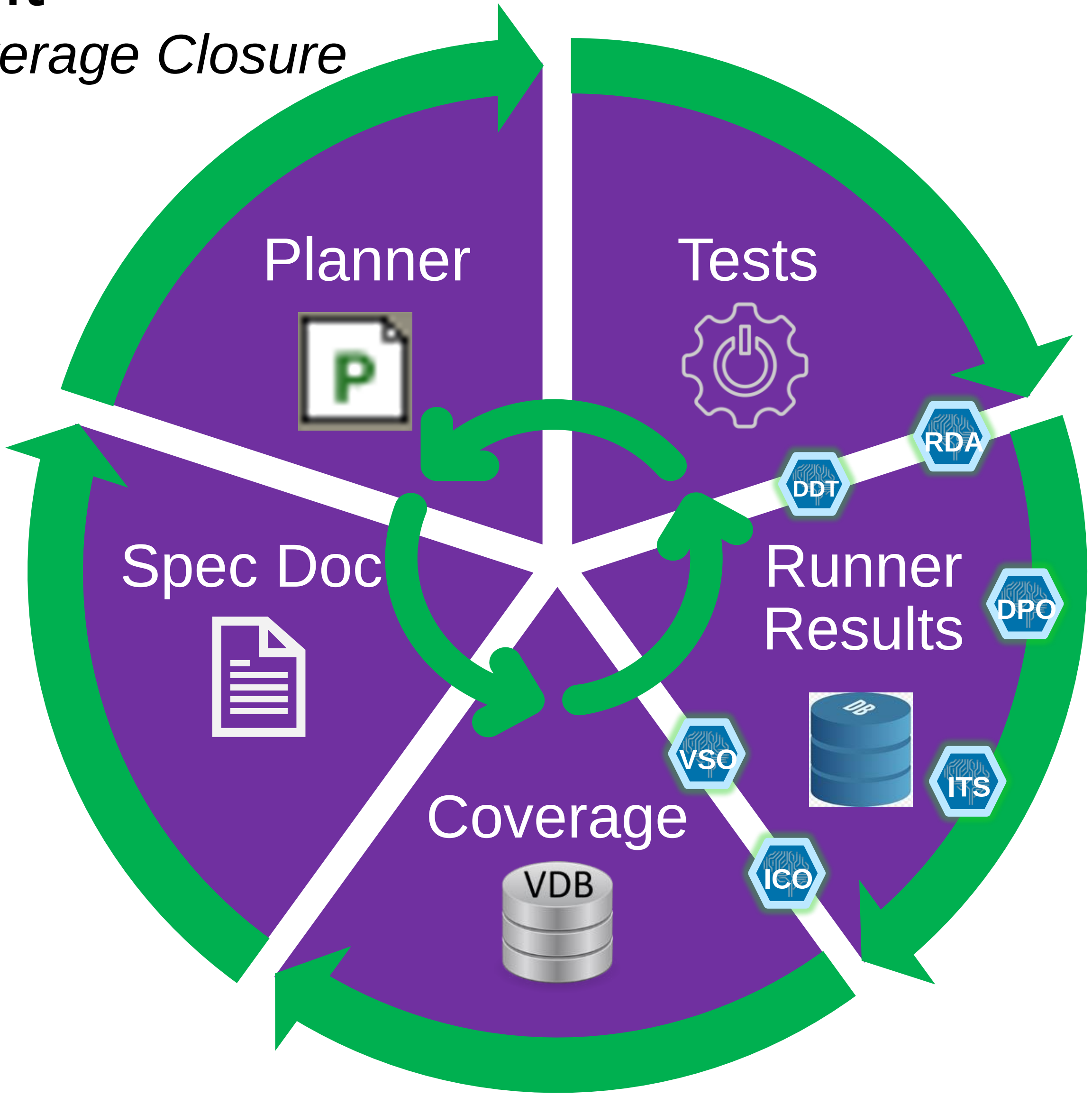
Debug

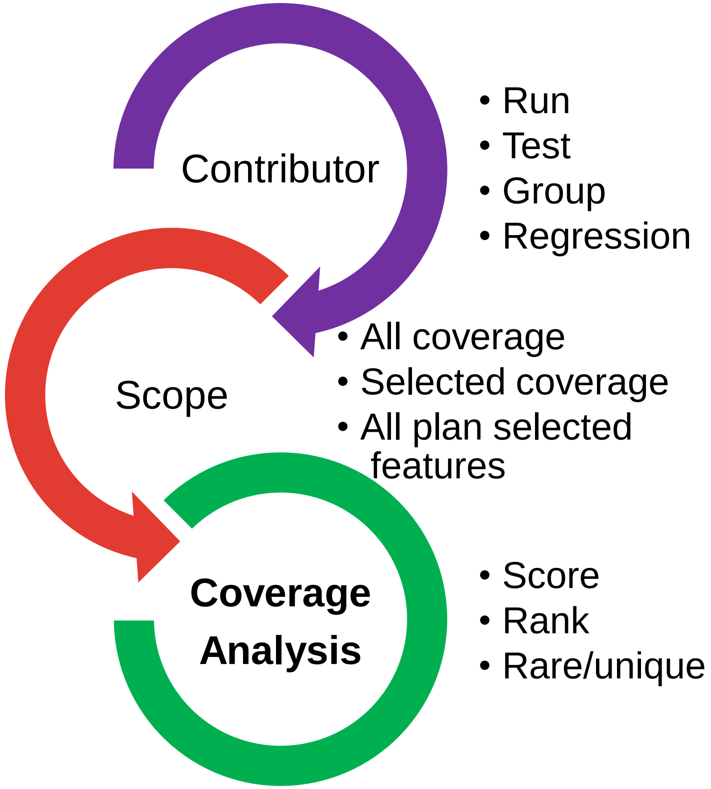
DDT

Unified Cockpit

Driving towards Coverage Closure

- All in one place GUI
- GUI layouts
- UI Customization
- Interconnections
- Coverage Results
- Coverage Analysis

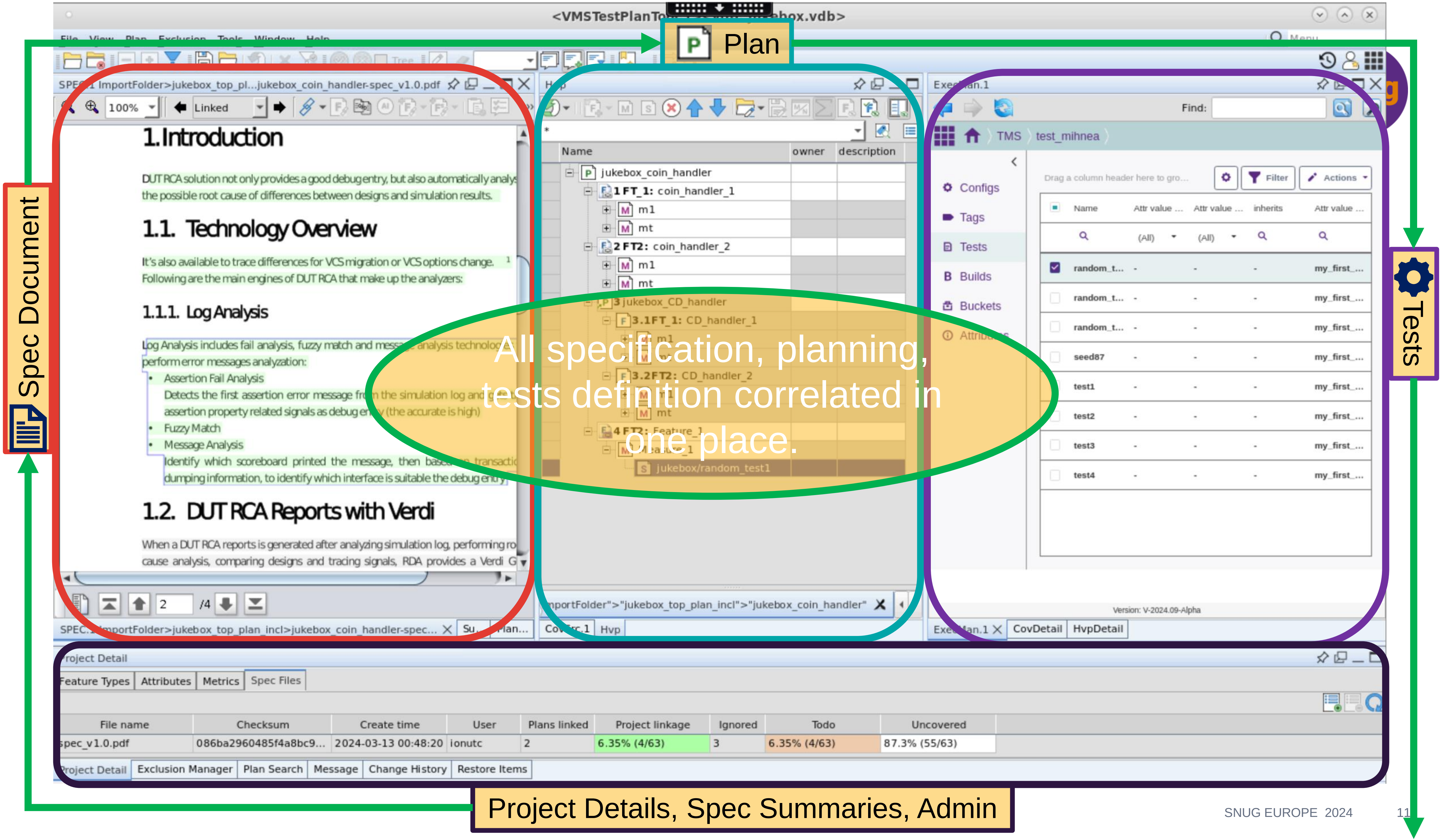




- Run
- Test
- Group
- Regression

- All coverage
- Selected coverage
- All plan selected features

- Score
- Rank
- Rare/unique



Plan
Spec Document
Tests
<VMSTestPlanTo…box.vdb>
File View Plan Exclusion Tools Window Help
SPEC.1 ImportFolder>jukebox_top_pl...jukebox_coin_handler-spec_v1.0.pdf
100%
Linked
2 /4
SPEC.1 ImportFolder>jukebox_top_plan_incl>jukebox_coin_handler-spec...
Su… Plan…
Hvp
Cov…rc.1 Hvp
…portFolder">"jukebox_top_plan_incl">"jukebox_coin_handler"
Exec…Plan.1
Find:
TMS
test_mihnea
Configs
Tags
Tests
Builds
Buckets
Attributes
Drag a column header here to gro...
Filter
Actions
Version: V-2024.09-Alpha
Exec…Plan.1 CovDetail HvpDetail

All specification, planning, tests definition correlated in one place.

1. Introduction

DUT RCA solution not only provides a good debug entry, but also automatically analys… the possible root cause of differences between designs and simulation results.

1.1. Technology Overview

It's also available to trace differences for VCS migration or VCS options change.[1] Following are the main engines of DUT RCA that make up the analyzers:

1.1.1. Log Analysis

Log Analysis includes fail analysis, fuzzy match and message analysis technolog… perform error messages analyzation:

- Assertion Fail Analysis
  Detects the first assertion error message from the simulation log and … assertion property related signals as debug entry (the accurate is high)
- Fuzzy Match
- Message Analysis
  Identify which scoreboard printed the message, then based on transacti… dumping information, to identify which interface is suitable the debug entry

1.2. DUT RCA Reports with Verdi

When a DUT RCA reports is generated after analyzing simulation log, performing ro… cause analysis, comparing designs and tracing signals, RDA provides a Verdi G…

Name	owner	description
P jukebox_coin_handler		
1 FT_1: coin_handler_1		
M m1		
M mt		
2 FT2: coin_handler_2		
M m1		
M mt		
P 3 jukebox_CD_handler		
F 3.1FT_1: CD_handler_1		
M m1		
M mt		
F 3.2FT2: CD_handler_2		
M m1		
M mt		
4 FT2: Feature_1		
M …ature_1		
S jukebox/random_test1		

☐	Name	Attr value …	Attr value …	inherits	Attr value …
		(All)	(All)		
☑	random_t…	-	-	-	my_first_…
☐	random_t…	-	-	-	my_first_…
☐	random_t…	-	-	-	my_first_…
☐	seed87	-	-	-	my_first_…
☐	test1	-	-	-	my_first_…
☐	test2	-	-	-	my_first_…
☐	test3	-	-	-	my_first_…
☐	test4	-	-	-	my_first_…

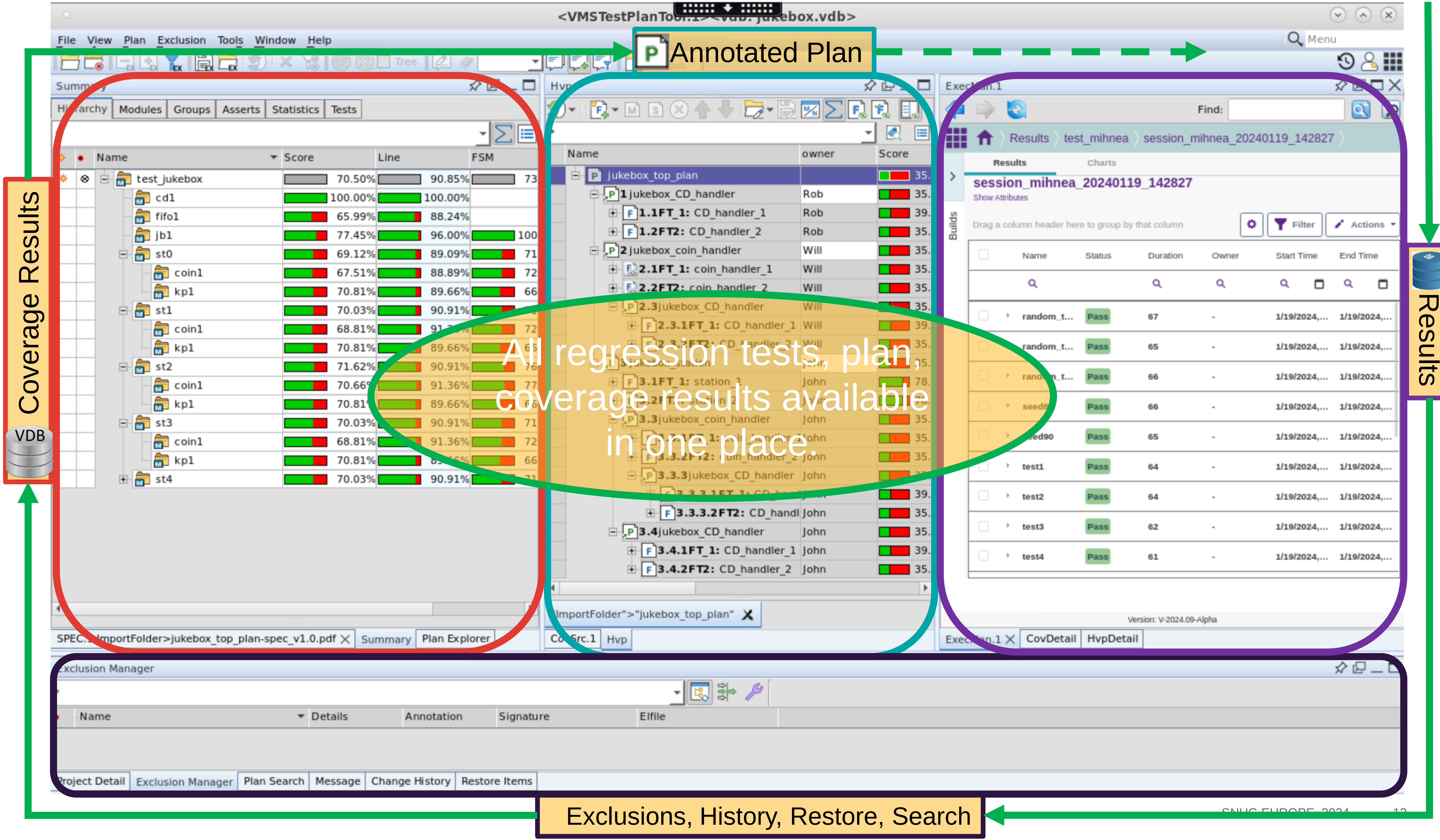
…roject Detail

Feature Types | Attributes | Metrics | Spec Files

File name	Checksum	Create time	User	Plans linked	Project linkage	Ignored	Todo	Uncovered
…spec_v1.0.pdf	086ba2960485f4a8bc9...	2024-03-13 00:48:20	ionutc	2	6.35% (4/63)	3	6.35% (4/63)	87.3% (55/63)

…roject Detail | Exclusion Manager | Plan Search | Message | Change History | Restore Items

Project Details, Spec Summaries, Admin



All regression tests, plan, coverage results available in one place.

Name	Score	Line	FSM
test_jukebox	70.50%	90.85%	73
cd1	100.00%	100.00%	
fifo1	65.99%	88.24%	
jb1	77.45%	96.00%	100
st0	69.12%	89.09%	71
coin1	67.51%	88.89%	72
kp1	70.81%	89.66%	66
st1	70.03%	90.91%	[illegible]
coin1	68.81%	91.3[illegible]	72
kp1	70.81%	89.66%	65
st2	71.62%	90.91%	76
coin1	70.66[illegible]	91.36%	77
kp1	70.81[illegible]	89.66%	66
st3	70.03%	90.91%	71
coin1	68.81%	91.36%	72
kp1	70.81%	[illegible]	66
st4	70.03%	90.91%	71

Name	owner	Score
jukebox_top_plan		35.
1 jukebox_CD_handler	Rob	35.
1.1FT_1: CD_handler_1	Rob	39.
1.2FT2: CD_handler_2	Rob	35.
2 jukebox_coin_handler	Will	35.
2.1FT_1: coin_handler_1	Will	35.
2.2FT2: coin_handler_2	Will	35.
2.3jukebox_CD_handler	Will	35.
2.3.1FT_1: CD_handler_1	Will	39.
2.3.2FT2: CD_handler_2	Will	35.
3 jukebox_station	John	35.
3.1FT_1: station_1	John	78.
3.2FT2: station_2	John	7[illegible]
3.3jukebox_coin_handler	John	35.
3.3.1FT_1: coin_handler_1	John	35.
3.3.2FT2: coin_handler_2	John	35.
3.3.3jukebox_CD_handler	John	[illegible]
3.3.3.1FT_1: CD_handl	John	39.
3.3.3.2FT2: CD_handl	John	35.
3.4jukebox_CD_handler	John	35.
3.4.1FT_1: CD_handler_1	John	39.
3.4.2FT2: CD_handler_2	John	35.

Results › test_mihnea › session_mihnea_20240119_142827

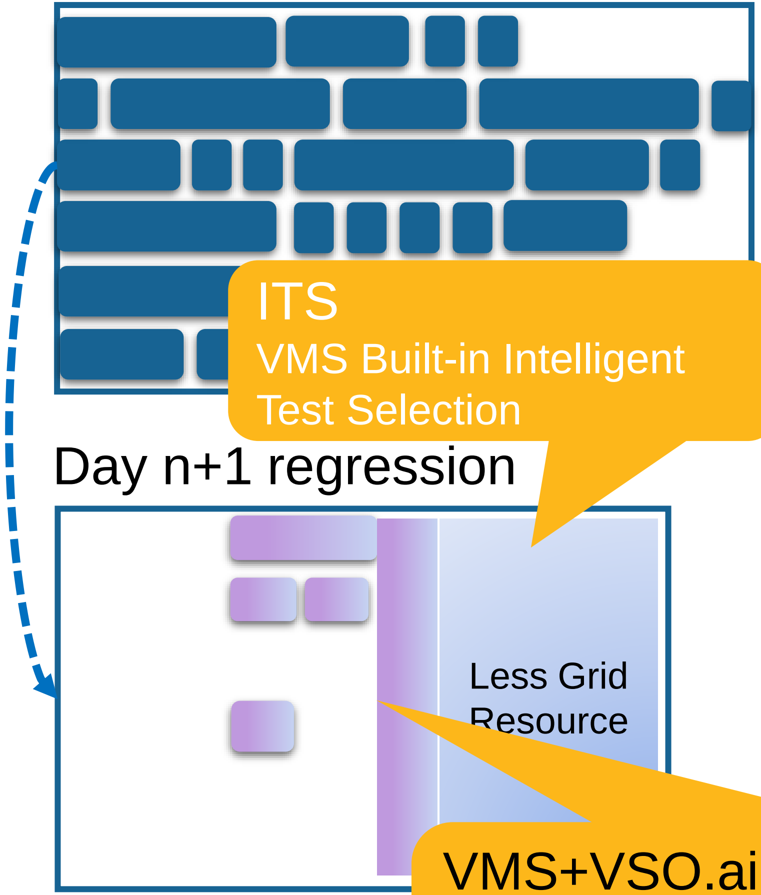
session_mihnea_20240119_142827

Name	Status	Duration	Owner	Start Time	End Time
random_t...	Pass	67	-	1/19/2024,...	1/19/2024,...
random_t...	Pass	65	-	1/19/2024,...	1/19/2024,...
random_t...	Pass	66	-	1/19/2024,...	1/19/2024,...
seed8[illegible]	Pass	66	-	1/19/2024,...	1/19/2024,...
[illegible]eed90	Pass	65	-	1/19/2024,...	1/19/2024,...
test1	Pass	64	-	1/19/2024,...	1/19/2024,...
test2	Pass	64	-	1/19/2024,...	1/19/2024,...
test3	Pass	62	-	1/19/2024,...	1/19/2024,...
test4	Pass	61	-	1/19/2024,...	1/19/2024,...

Name	Details	Annotation	Signature	Elfile

Optimized Regression with VMS+VSO.ai

Day 0 regression



ITS
VMS Built-in Intelligent Test Selection

Day n+1 regression

Less Grid Resource

VMS+VSO.ai Native Integration

VCS compile

- Static
- Dynamic
- User Hints
- ML

VSO.ai Coverage Inference

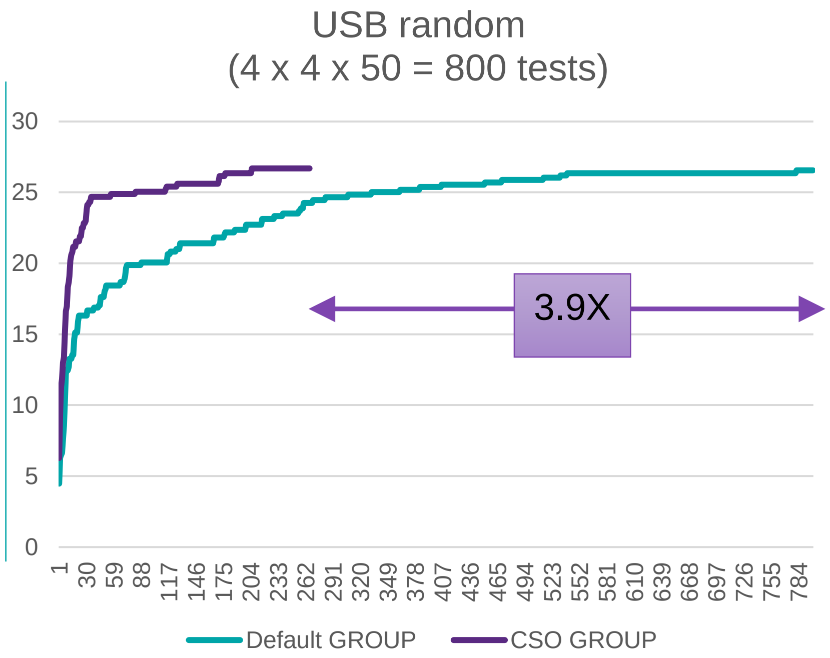
% simv run +testN

VSO.ai Coverage Directed Solver

VDB coverage database

Coverage Report

VSO.ai Illegal cov and RCA report



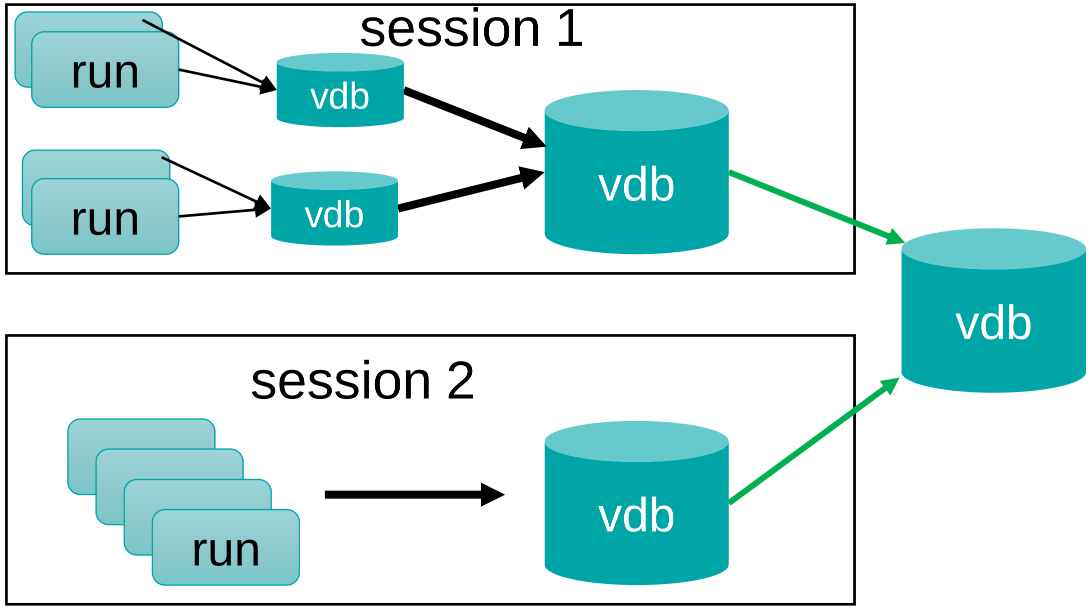
VMS+VSO.ai

- Optimize out tests/seeds
- Control seeds and runtime args
- Prioritize and extend rare tests

Standalone aggregation or integrated inside Runner

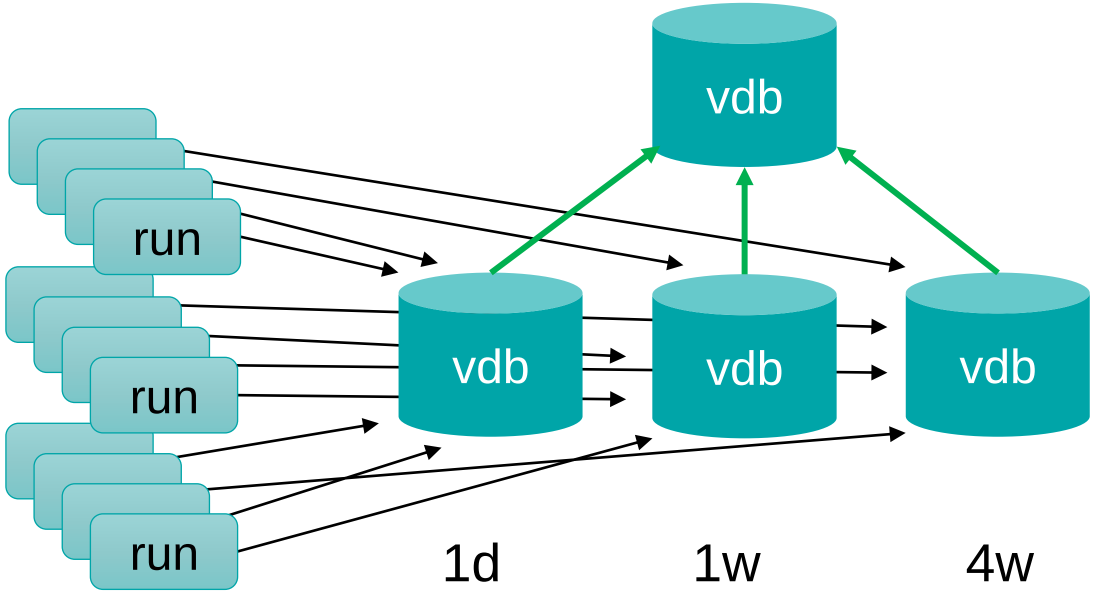
Runner Coverage Merging

- Merges are tied to specific build and runs
- Final merged VDB and session report
- Session VDBs can be combined separately



Standalone Coverage Merging

- VDBs come from multiple builds and runs
- Continuous merge incoming VDBs
- Tree combines results at higher levels



VMS debug automation

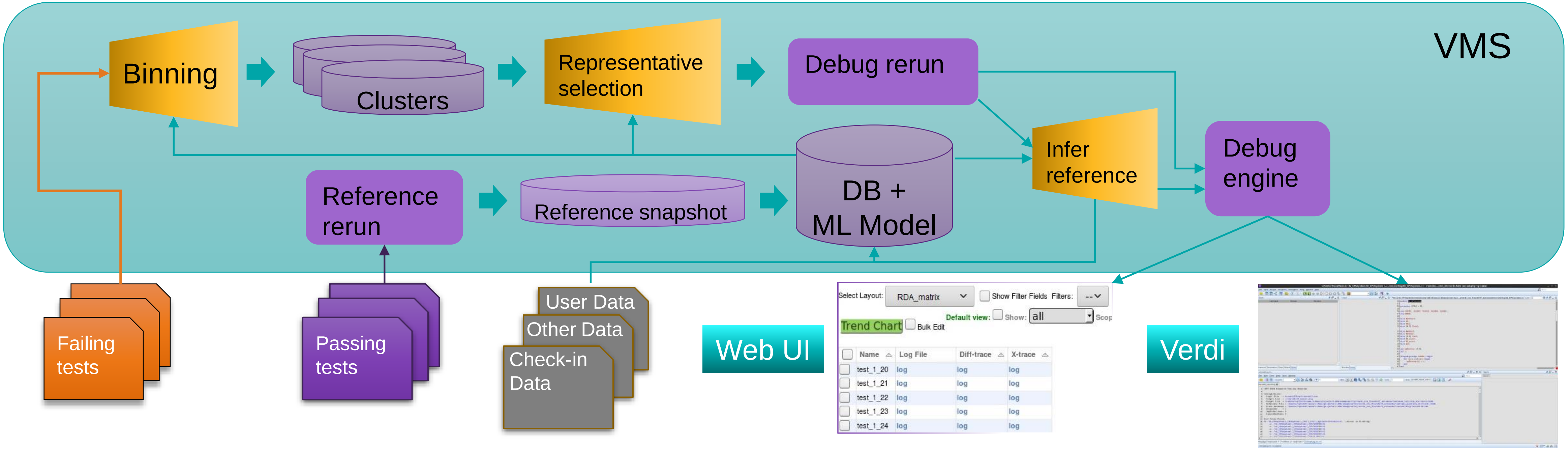
Native support for debug automation and root cause analysis in regression

Value Proposition

- Native flow for Binning, Debug, Rerun and Root cause analysis flows
- Open API for custom Binning and Root cause algorithms (e.g. fsdb based, check-in based)
- Open API for DB and ML models
- Reference capture and inference
- Easy enablement in regression flows

Flow

- Define Debug engines and policies in easy and generic way in the config file.
- Categorize failing tests into clusters using Error patterns and ML.
- Infer representative tests per cluster for debug.
- Make reference snapshots of predefined passing tests.
  - Infer reference snapshot or re-generate it on the spot.
- Run Debug engines as natively embedded step in VMS flow.
- Debug engines results ready at end of the session in web browser UI or Verdi.

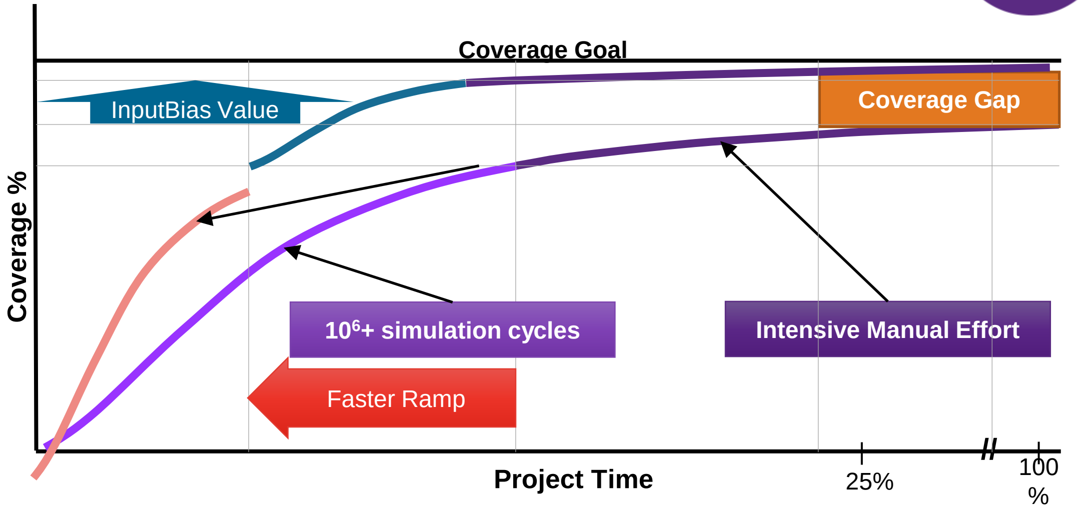


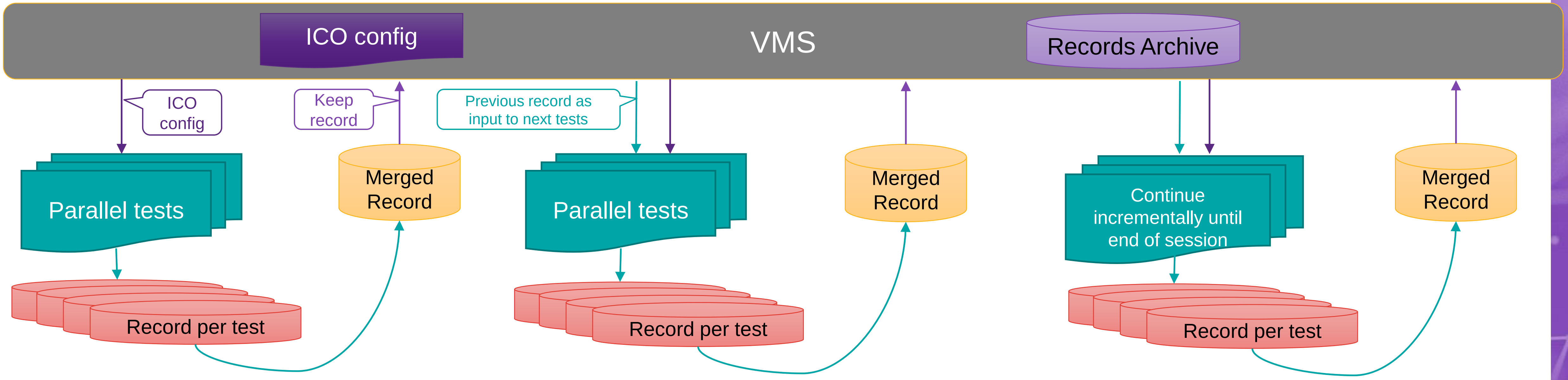
	Name	Log File	Diff-trace	X-trace
□	test_1_20	log	log	log
□	test_1_21	log	log	log
□	test_1_22	log	log	log
□	test_1_23	log	log	log
□	test_1_24	log	log	log

Native Integration of Intelligent Coverage Optimization (ICO)

Enabling constraints biasing on full regression

- Achieve higher, faster coverage
- Catch more bugs
- Leverage VMS managed infrastructure
- Use simple interface
- Merge server scalable technology
- Run parent session
- Provides consistent debug and retry
- Reports
- Grade results



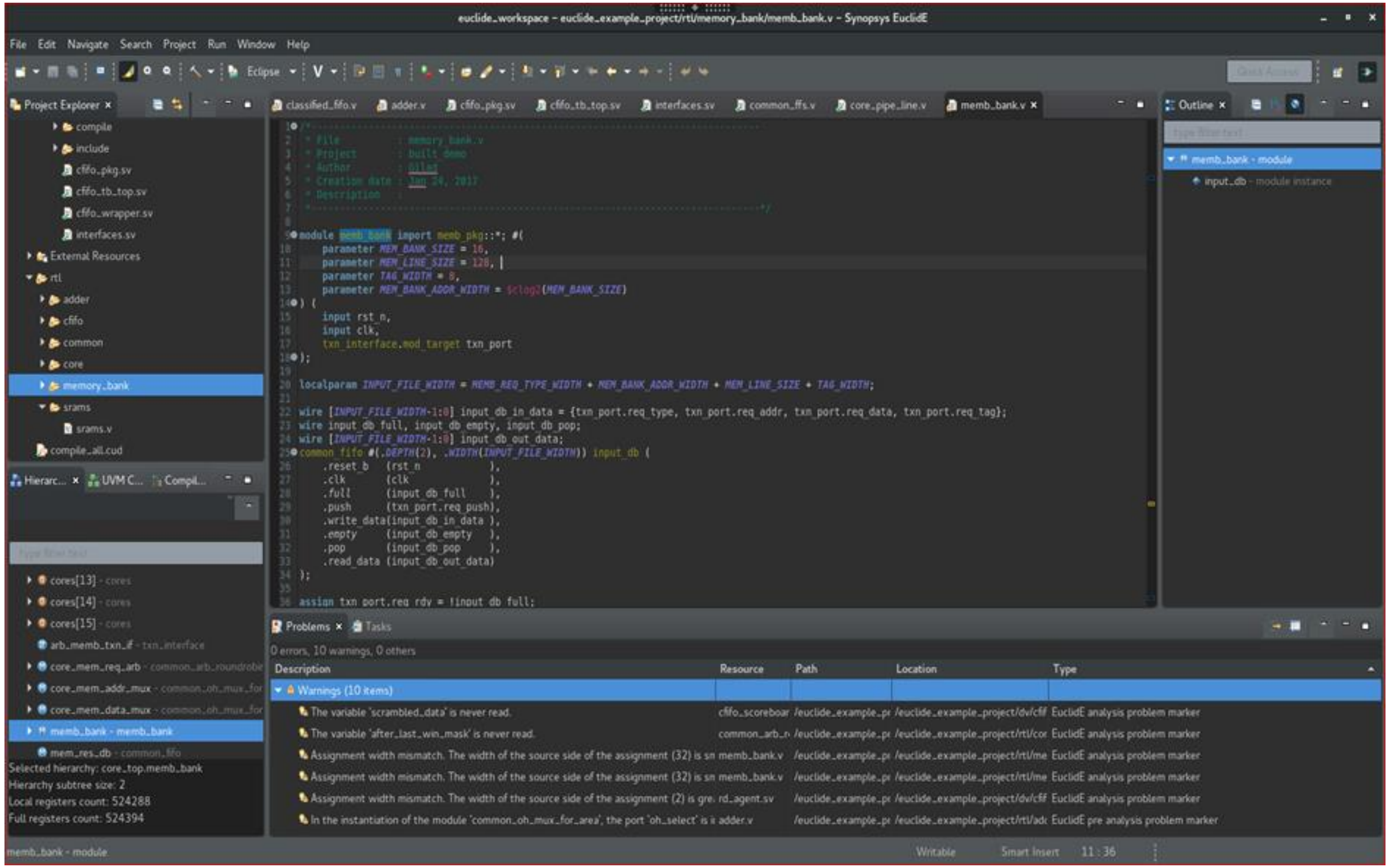


Euclide

Verdi Integrated Design Environment (IDE)

Euclide IDE

- On-the-fly SVTB linter and design checks
- Content assistance
- Hover information
- Hyperlinks
- Design hierarchy tree
- UVM component hierarchy and more
- Task management



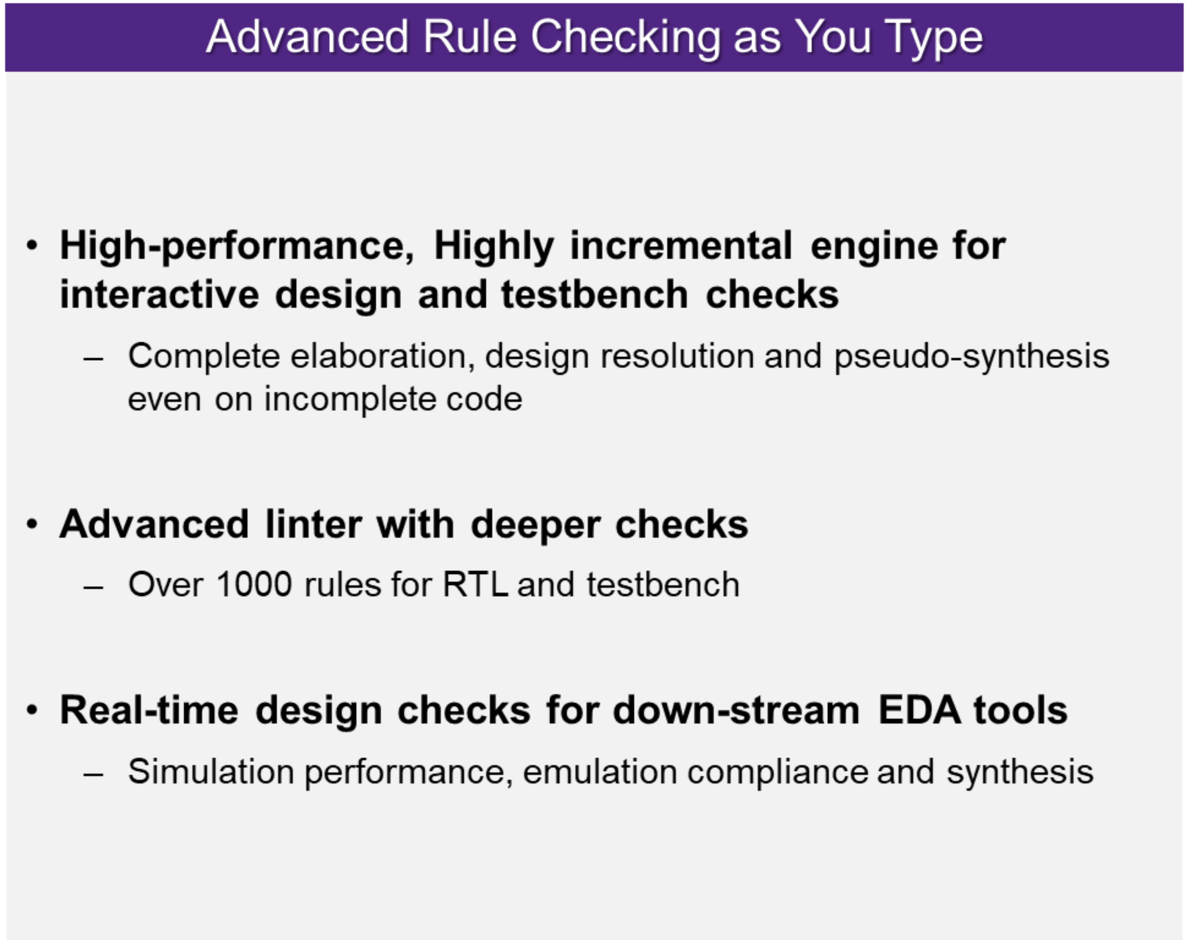
Selected hierarchy: core_top.memb_bank
Hierarchy subtree size: 2
Local registers count: 524288
Full registers count: 524394

```
/*-------------------------------------------------------------------------
 * File          : memory_bank.v
 * Project       : built_demo
 * Author        : dillat
 * Creation date : Jan 24, 2017
 * Description   :
 *-----------------------------------------------------------------------*/

module memb_bank import memb_pkg::*; #(
    parameter MEM_BANK_SIZE = 16,
    parameter MEM_LINE_SIZE = 128,
    parameter TAG_WIDTH = 8,
    parameter MEM_BANK_ADDR_WIDTH = $clog2(MEM_BANK_SIZE)
) (
    input rst_n,
    input clk,
    txn_interface.mod_target txn_port
);

localparam INPUT_FILE_WIDTH = MEMB_REQ_TYPE_WIDTH + MEM_BANK_ADDR_WIDTH + MEM_LINE_SIZE + TAG_WIDTH;

wire [INPUT_FILE_WIDTH-1:0] input_db_in_data = {txn_port.req_type, txn_port.req_addr, txn_port.req_data, txn_port.req_tag};
wire input_db_full, input_db_empty, input_db_pop;
wire [INPUT_FILE_WIDTH-1:0] input_db_out_data;
common_fifo #(.DEPTH(2), .WIDTH(INPUT_FILE_WIDTH)) input_db (
    .reset_b   (rst_n           ),
    .clk       (clk             ),
    .full      (input_db_full   ),
    .push      (txn_port.req_push),
    .write_data(input_db_in_data ),
    .empty     (input_db_empty  ),
    .pop       (input_db_pop    ),
    .read_data (input_db_out_data)
);

assign txn_port.req_rdy = !input_db_full;
```

0 errors, 10 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (10 items)				
The variable 'scrambled_data' is never read.	cfifo_scoreboar	/euclide_example_pr	/euclide_example_project/dv/cfif	EuclidE analysis problem marker
The variable 'after_last_win_mask' is never read.	common_arb_r	/euclide_example_pr	/euclide_example_project/rtl/cor	EuclidE analysis problem marker
Assignment width mismatch. The width of the source side of the assignment (32) is sn	memb_bank.v	/euclide_example_pr	/euclide_example_project/rtl/me	EuclidE analysis problem marker
Assignment width mismatch. The width of the source side of the assignment (32) is sn	memb_bank.v	/euclide_example_pr	/euclide_example_project/rtl/me	EuclidE analysis problem marker
Assignment width mismatch. The width of the source side of the assignment (2) is gre	rd_agent.sv	/euclide_example_pr	/euclide_example_project/dv/cfif	EuclidE analysis problem marker
In the instantiation of the module 'common_oh_mux_for_area', the port 'oh_select' is i	adder.v	/euclide_example_pr	/euclide_example_project/rtl/adc	EuclidE pre analysis problem marker

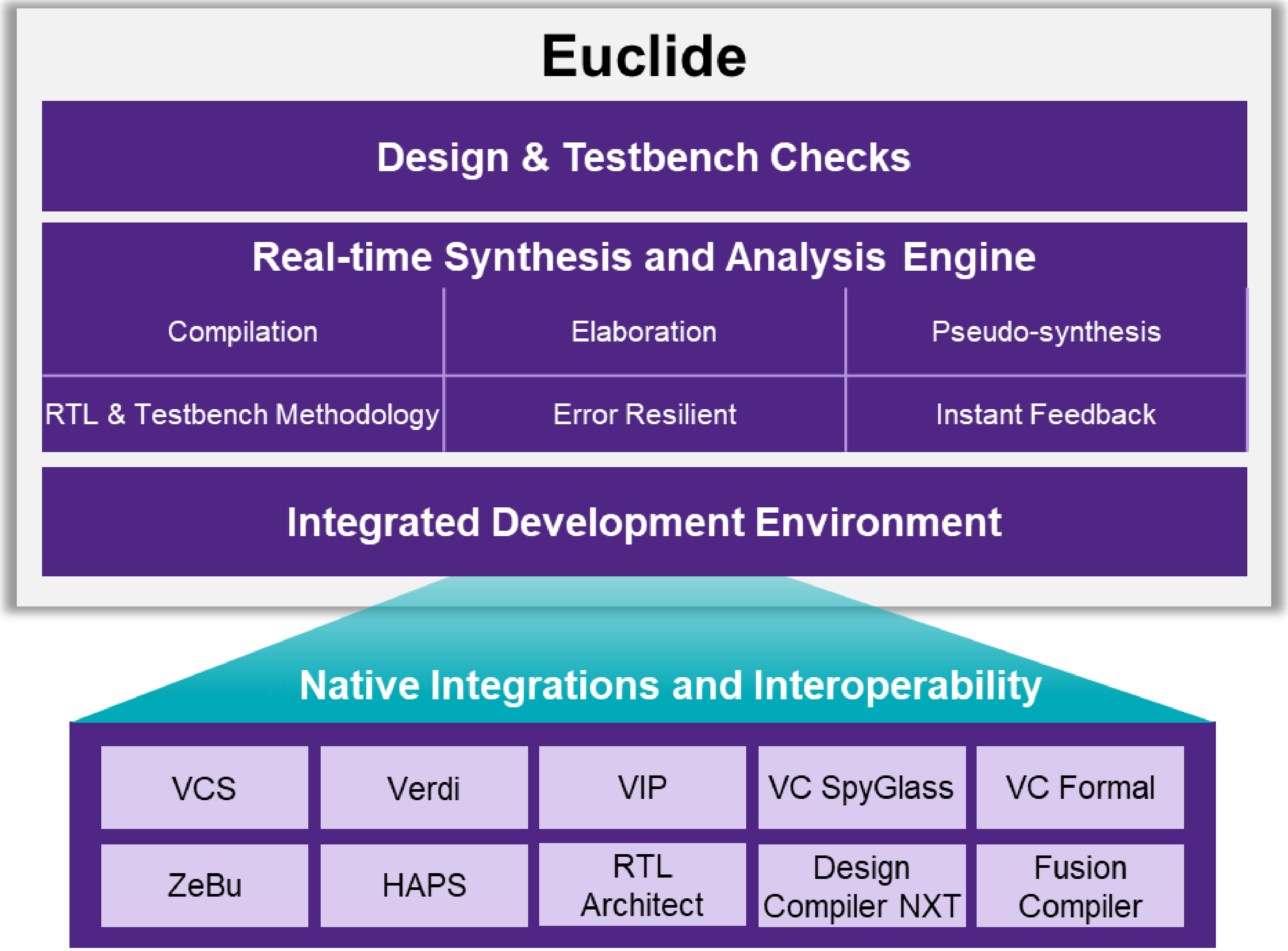
Euclide IDE



Advanced Rule Checking as You Type

- **High-performance, Highly incremental engine for interactive design and testbench checks**
  - Complete elaboration, design resolution and pseudo-synthesis even on incomplete code
- **Advanced linter with deeper checks**
  - Over 1000 rules for RTL and testbench
- **Real-time design checks for down-stream EDA tools**
  - Simulation performance, emulation compliance and synthesis

Under the hood



Euclide

Design & Testbench Checks

Real-time Synthesis and Analysis Engine

Compilation

Elaboration

Pseudo-synthesis

RTL & Testbench Methodology

Error Resilient

Instant Feedback

Integrated Development Environment

Native Integrations and Interoperability

VCS

Verdi

VIP

VC SpyGlass

VC Formal

ZeBu

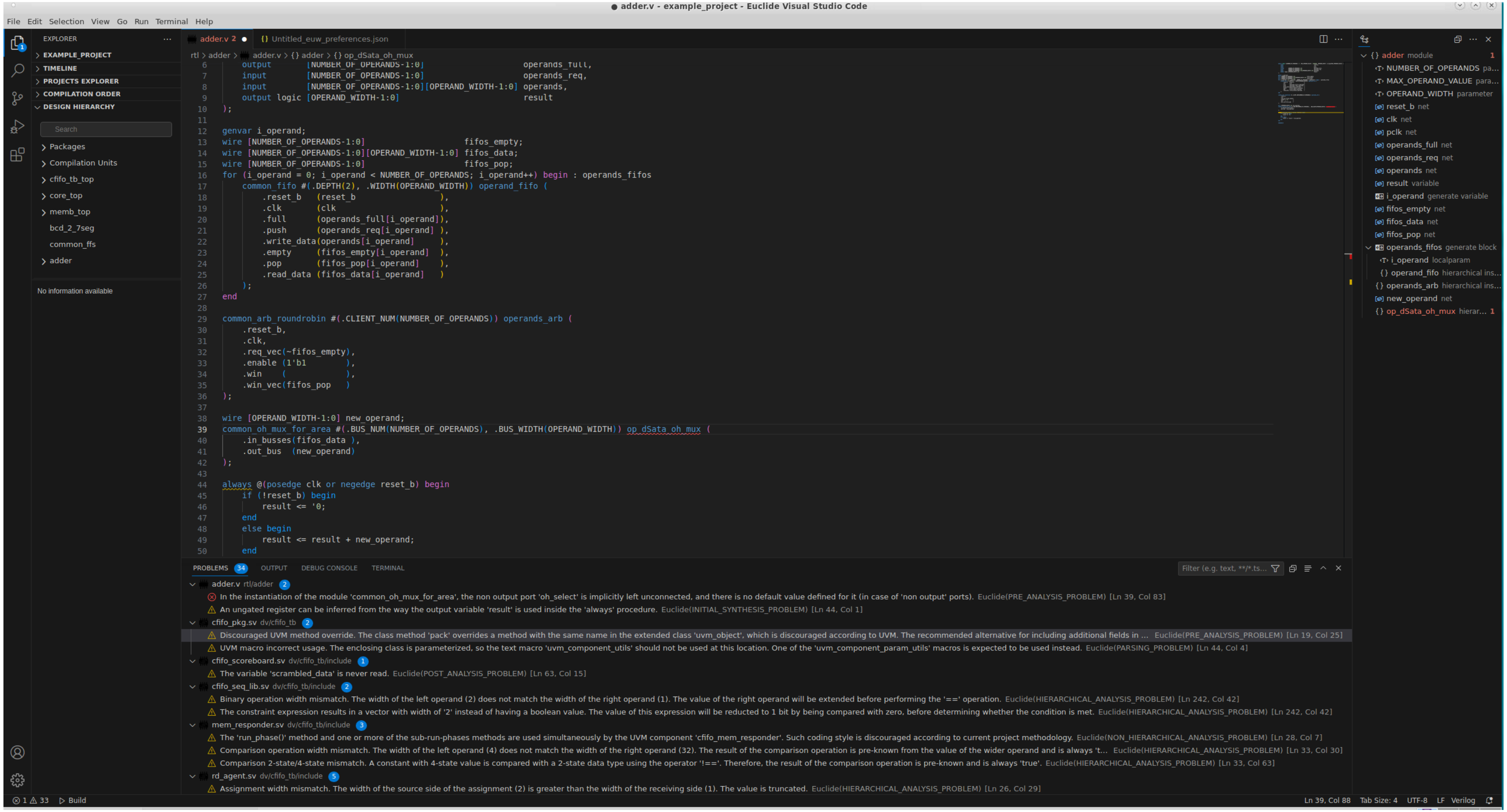
HAPS

RTL Architect

Design Compiler NXT

Fusion Compiler

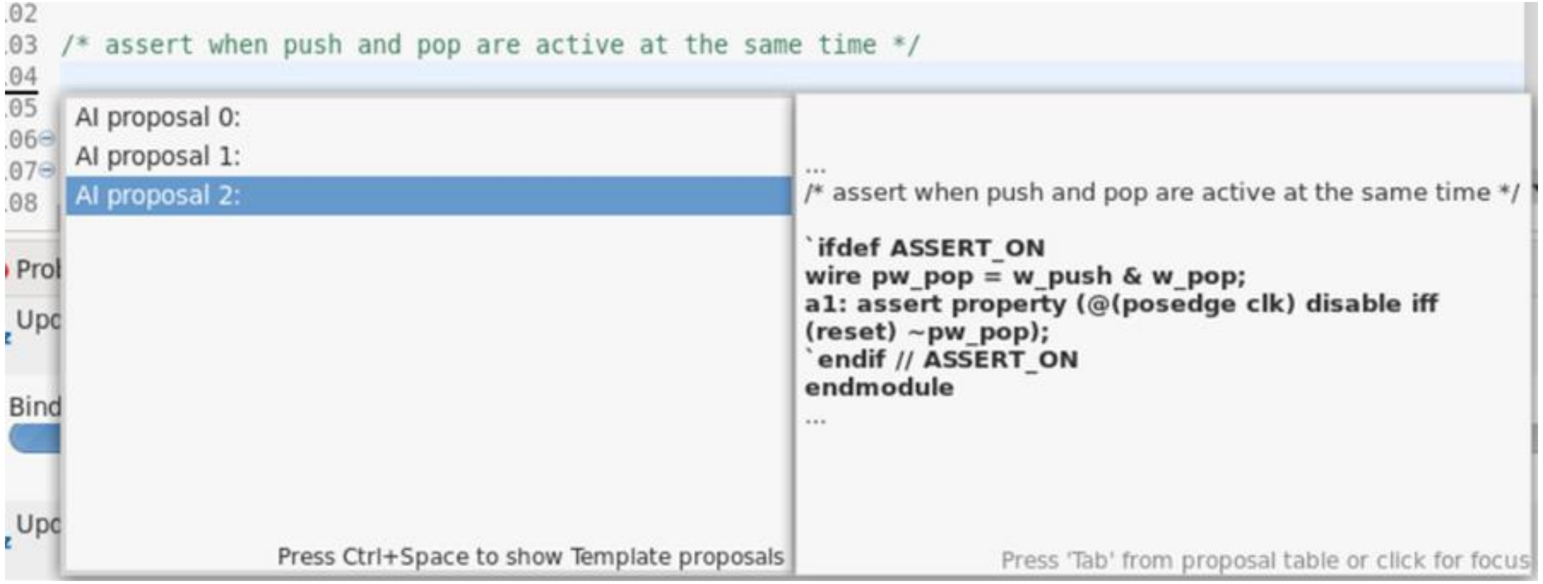
There is also VS Code



```
    output       [NUMBER_OF_OPERANDS-1:0]                    operands_full,
    input        [NUMBER_OF_OPERANDS-1:0]                    operands_req,
    input        [NUMBER_OF_OPERANDS-1:0][OPERAND_WIDTH-1:0] operands,
    output logic [OPERAND_WIDTH-1:0]                         result
);

genvar i_operand;
wire [NUMBER_OF_OPERANDS-1:0]                    fifos_empty;
wire [NUMBER_OF_OPERANDS-1:0][OPERAND_WIDTH-1:0] fifos_data;
wire [NUMBER_OF_OPERANDS-1:0]                    fifos_pop;
for (i_operand = 0; i_operand < NUMBER_OF_OPERANDS; i_operand++) begin : operands_fifos
    common_fifo #(.DEPTH(2), .WIDTH(OPERAND_WIDTH)) operand_fifo (
        .reset_b   (reset_b                 ),
        .clk       (clk                     ),
        .full      (operands_full[i_operand]),
        .push      (operands_req[i_operand] ),
        .write_data(operands[i_operand]     ),
        .empty     (fifos_empty[i_operand]  ),
        .pop       (fifos_pop[i_operand]    ),
        .read_data (fifos_data[i_operand]   )
    );
end

common_arb_roundrobin #(.CLIENT_NUM(NUMBER_OF_OPERANDS)) operands_arb (
    .reset_b,
    .clk,
    .req_vec(~fifos_empty),
    .enable (1'b1        ),
    .win    (            ),
    .win_vec(fifos_pop   )
);

wire [OPERAND_WIDTH-1:0] new_operand;
common_oh_mux_for_area #(.BUS_NUM(NUMBER_OF_OPERANDS), .BUS_WIDTH(OPERAND_WIDTH)) op_dSata_oh_mux (
    .in_busses(fifos_data ),
    .out_bus  (new_operand)
);

always @(posedge clk or negedge reset_b) begin
    if (!reset_b) begin
        result <= '0;
    end
    else begin
        result <= result + new_operand;
    end
```

```
adder.v rtl/adder 2
  ⊗ In the instantiation of the module 'common_oh_mux_for_area', the non output port 'oh_select' is implicitly left unconnected, and there is no default value defined for it (in case of 'non output' ports). Euclide(PRE_ANALYSIS_PROBLEM) [Ln 39, Col 83]
  ⚠ An ungated register can be inferred from the way the output variable 'result' is used inside the 'always' procedure. Euclide(INITIAL_SYNTHESIS_PROBLEM) [Ln 44, Col 1]
cfifo_pkg.sv dv/cfifo_tb 2
  ⚠ Discouraged UVM method override. The class method 'pack' overrides a method with the same name in the extended class 'uvm_object', which is discouraged according to UVM. The recommended alternative for including additional fields in ...  Euclide(PRE_ANALYSIS_PROBLEM) [Ln 19, Col 25]
  ⚠ UVM macro incorrect usage. The enclosing class is parameterized, so the text macro 'uvm_component_utils' should not be used at this location. One of the 'uvm_component_param_utils' macros is expected to be used instead. Euclide(PARSING_PROBLEM) [Ln 44, Col 4]
cfifo_scoreboard.sv dv/cfifo_tb/include 1
  ⚠ The variable 'scrambled_data' is never read. Euclide(POST_ANALYSIS_PROBLEM) [Ln 63, Col 15]
cfifo_seq_lib.sv dv/cfifo_tb/include 2
  ⚠ Binary operation width mismatch. The width of the left operand (2) does not match the width of the right operand (1). The value of the right operand will be extended before performing the '==' operation. Euclide(HIERARCHICAL_ANALYSIS_PROBLEM) [Ln 242, Col 42]
  ⚠ The constraint expression results in a vector with width of '2' instead of having a boolean value. The value of this expression will be reducted to 1 bit by being compared with zero, before determining whether the condition is met. Euclide(HIERARCHICAL_ANALYSIS_PROBLEM) [Ln 242, Col 42]
mem_responder.sv dv/cfifo_tb/include 3
  ⚠ The 'run_phase()' method and one or more of the sub-run-phases methods are used simultaneously by the UVM component 'cfifo_mem_responder'. Such coding style is discouraged according to current project methodology. Euclide(NON_HIERARCHICAL_ANALYSIS_PROBLEM) [Ln 28, Col 7]
  ⚠ Comparison operation width mismatch. The width of the left operand (4) does not match the width of the right operand (32). The result of the comparison operation is pre-known from the value of the wider operand and is always 't...  Euclide(HIERARCHICAL_ANALYSIS_PROBLEM) [Ln 33, Col 30]
  ⚠ Comparison 2-state/4-state mismatch. A constant with 4-state value is compared with a 2-state data type using the operator '!=='. Therefore, the result of the comparison operation is pre-known and is always 'true'. Euclide(HIERARCHICAL_ANALYSIS_PROBLEM) [Ln 33, Col 63]
rd_agent.sv dv/cfifo_tb/include 5
  ⚠ Assignment width mismatch. The width of the source side of the assignment (2) is greater than the width of the receiving side (1). The value is truncated. Euclide(HIERARCHICAL_ANALYSIS_PROBLEM) [Ln 26, Col 29]
```

A look ahead: AI features



```
/* assert when push and pop are active at the same time */
```

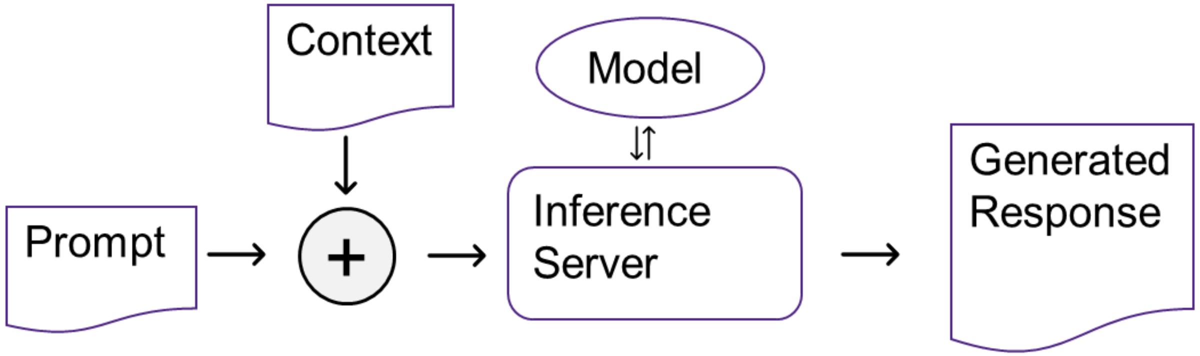
AI proposal 0:
AI proposal 1:
AI proposal 2:
Prol
Upc
Bind
Upc

```
...
/* assert when push and pop are active at the same time */

`ifdef ASSERT_ON
wire pw_pop = w_push & w_pop;
a1: assert property (@(posedge clk) disable iff
(reset) ~pw_pop);
`endif // ASSERT_ON
endmodule
...
```

Press Ctrl+Space to show Template proposals
Press 'Tab' from proposal table or click for focus

- Euclide GenAI client features:
  - AI code generation
  - Code explainer and documentation generator
  - Usage statistics collection
  - Compiled project context

- Euclide GenAI backend/model features:
  - On-prem LLM specialized in the domain of RTL design and verification
  - Clients to easily fine-tune Synopsys model using their data for training
  - Evaluation framework to benchmark different models



Context

Model

Prompt

+

Inference Server

Generated Response

Summary

Summary

- **Next-Gen Verdi** lets you avoid manual regression debug is tedious
  - Automate it with AI and advanced RCA technologies to debug any failing simulations

- Use **VC Execution Manager** (VMS) as a Unified Cockpit to create verification plans, manage simulation, gather coverage data and analyze date, optimize regression TAT and achieve faster coverage closure
  - Available with Verdi (2023.12)

- Create correct code by construction by using **Euclide**
  - Available with Verdi (2023.12)

THANK YOU

Our
Technology,
Your
Innovation™