



Agentic AI for Next-Generation Chip Verification

Verisium Powered by JedAI

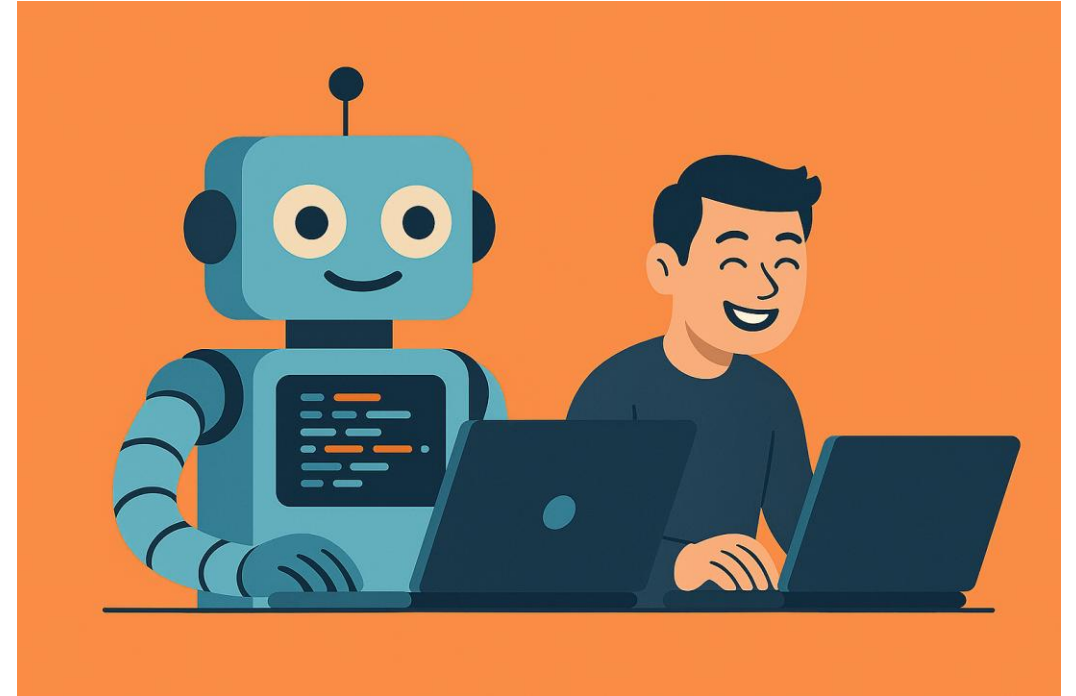
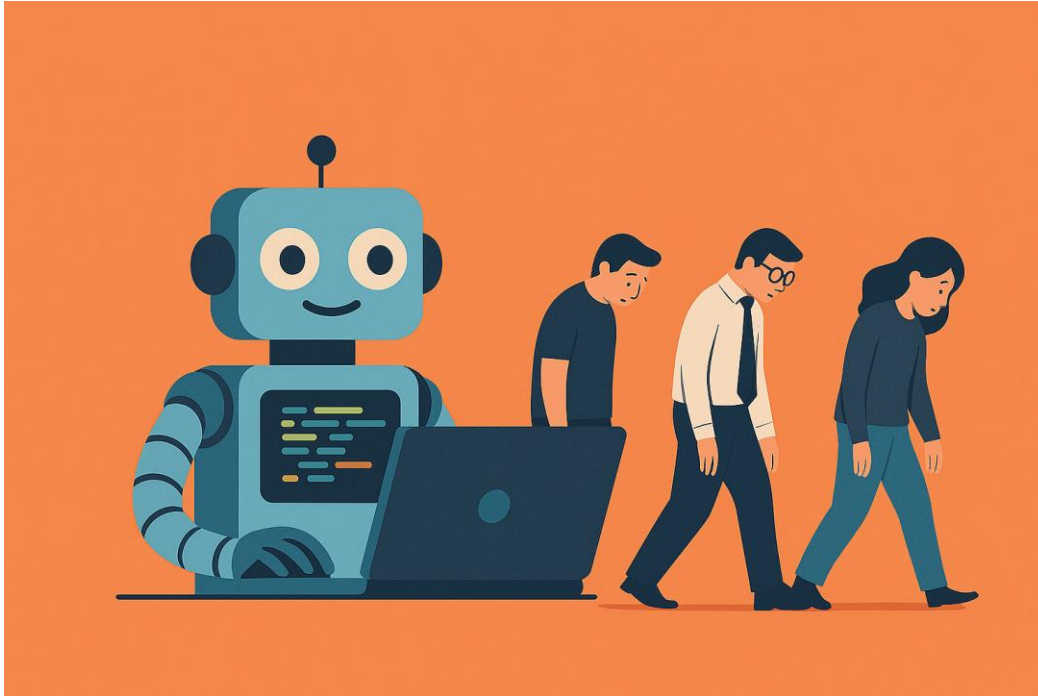
Anika Sunda – Director Product Management, System Verification Group

December 16, 2025

 **cā dence**[®]

Engineers won't lose their jobs to AI.

They will lose their jobs to another engineer using AI. – Jensen Huang



AI Coding – Making the world a better place?

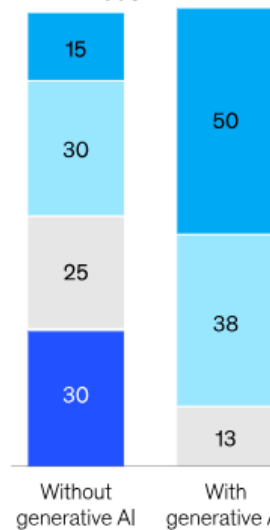
- Automate tedious tasks
 - Boiler plate and syntax
 - Unit tests and edge cases
 - Debugging
 - Documentation
- Focus engineering time on creative problem-solving
- Improve morale and engagement – “Happy bees make sweeter honey.”

Generative AI tools have potential to improve the developer experience.

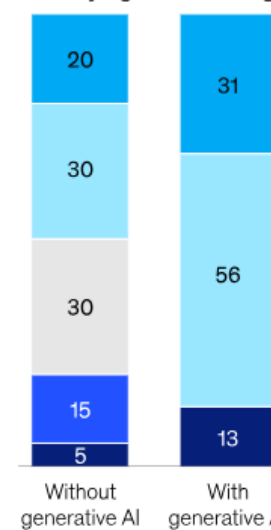
Agreement with statement,
% of respondents

Strongly disagree Somewhat disagree Neither agree or disagree Somewhat agree Strongly agree

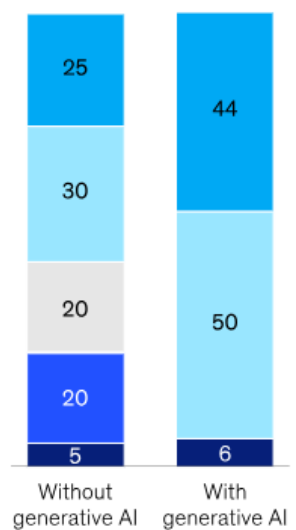
I felt happy



I was able to focus on satisfying and meaningful work



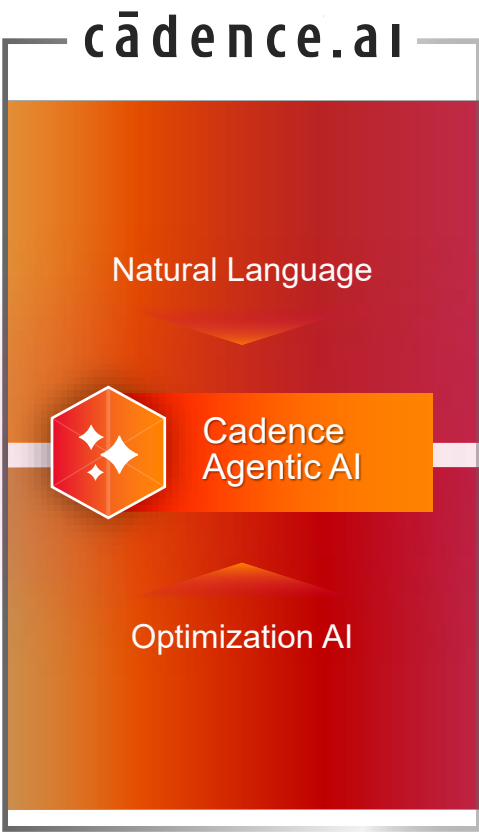
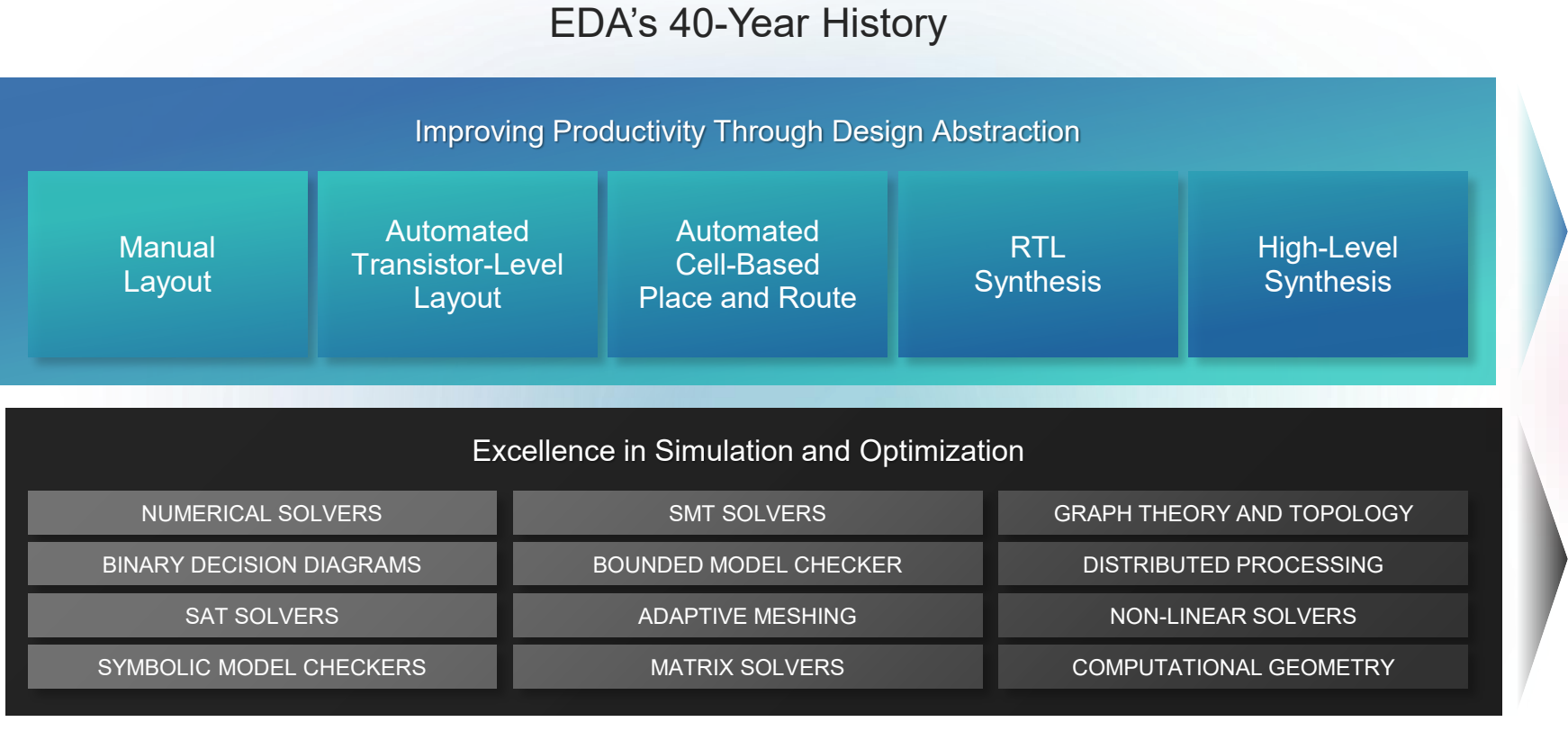
I was in a 'flow' state



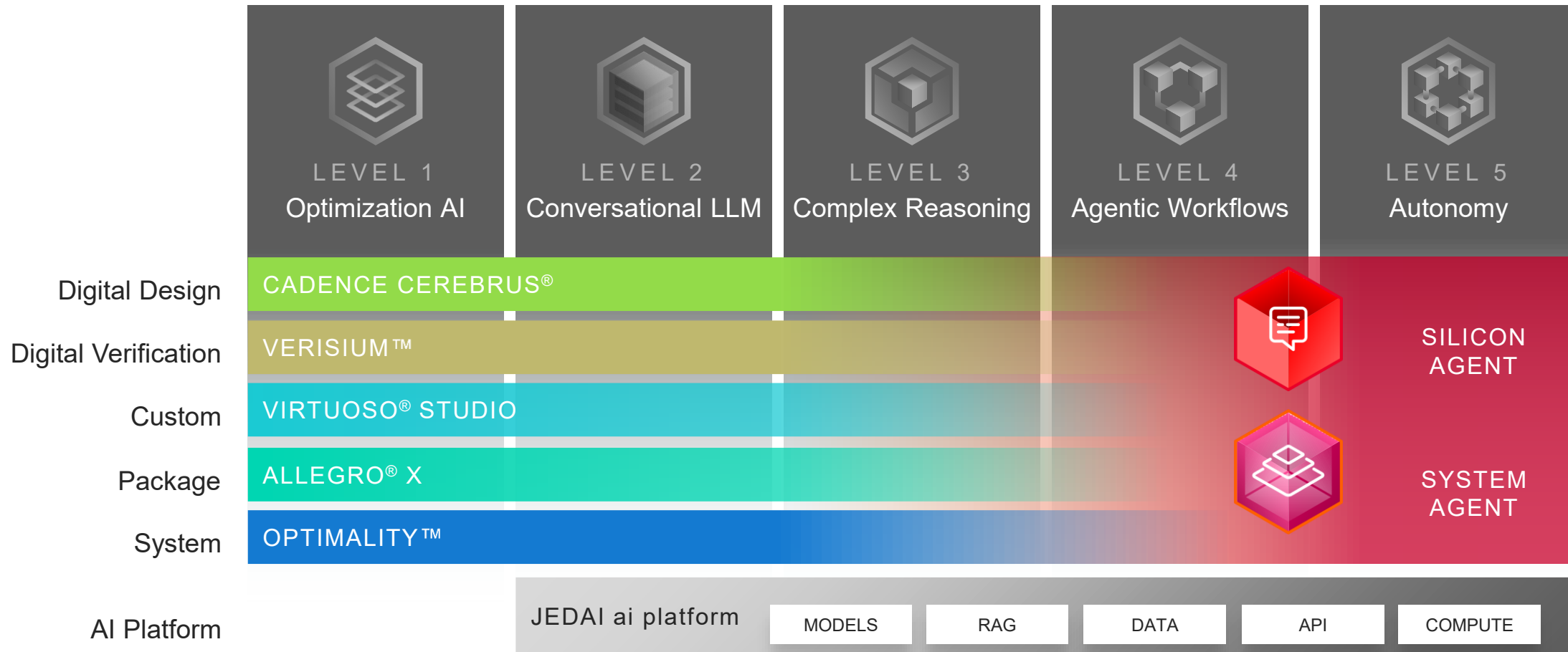
Note: Figures may not sum to 100%, because of rounding.

McKinsey & Company

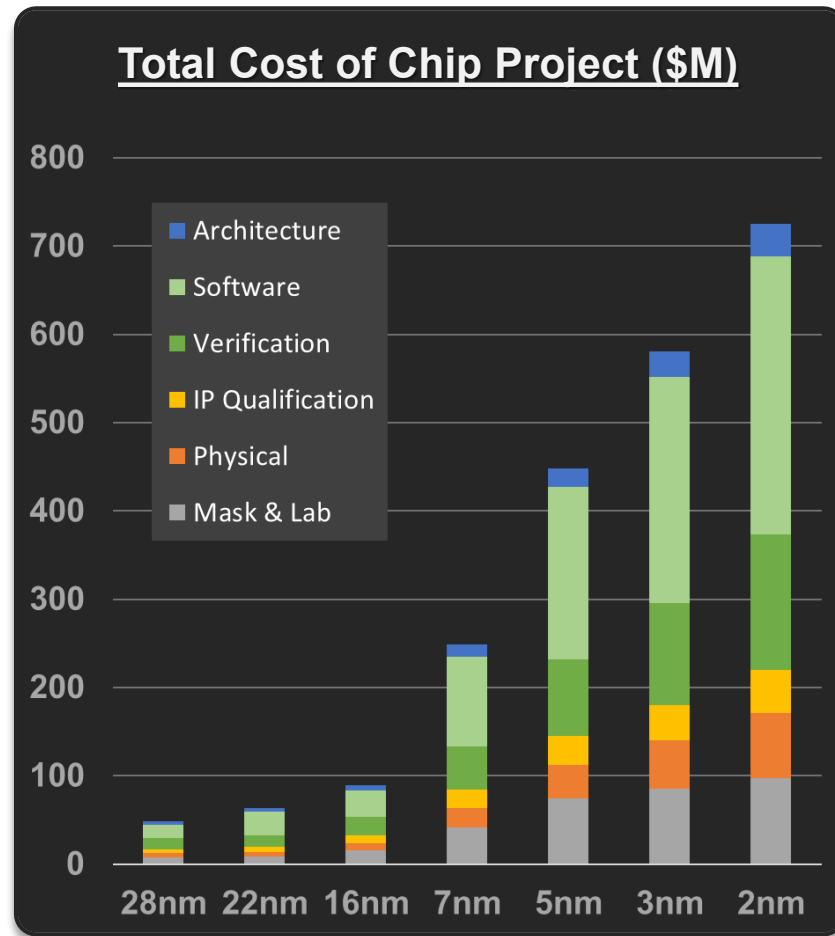
Agentic AI: Optimize Engines & Optimize Engineers



The Journey to Autonomous Design



Verification is Still the Infinite Problem



IBS, Global Semiconductor Industry Service Report

- Verification largest cost in IC dev
- Advanced node and 3D-IC means more gates, more states, more potential bugs
- Dev schedules continue to tighten
- Engineering resource gap is growing

Agentic AI Journey – Level1 - Optimization AI

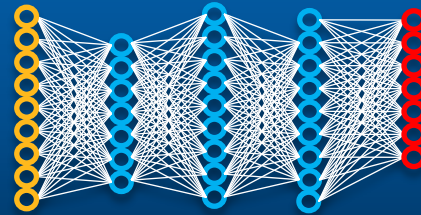
Digital Implementation (Cerebrus)



Reinforcement Learning

Node	Productivity	PPA
4nm	5x	8% leakage
4nm	5x	10% leakage
6nm	5x	20% PPA
5nm	5x	14% Fmax
7nm	5x	8% power
5nm	5x	15% power
N12	4x	13% PPA
5nm	4x	25% power

Digital Simulation (SimAI)



Reinforcement Learning

Design Type	Speed Up
Modem IP	20x
Mobile SoC	7x
AI IP	8x
AI IP	3x
Memory IP	5x
Base Station	18x
NOC IP	5x
Automotive SoC	5x
CPU IP	3x

Digital Formal Verification (Formal AI - Jasper)

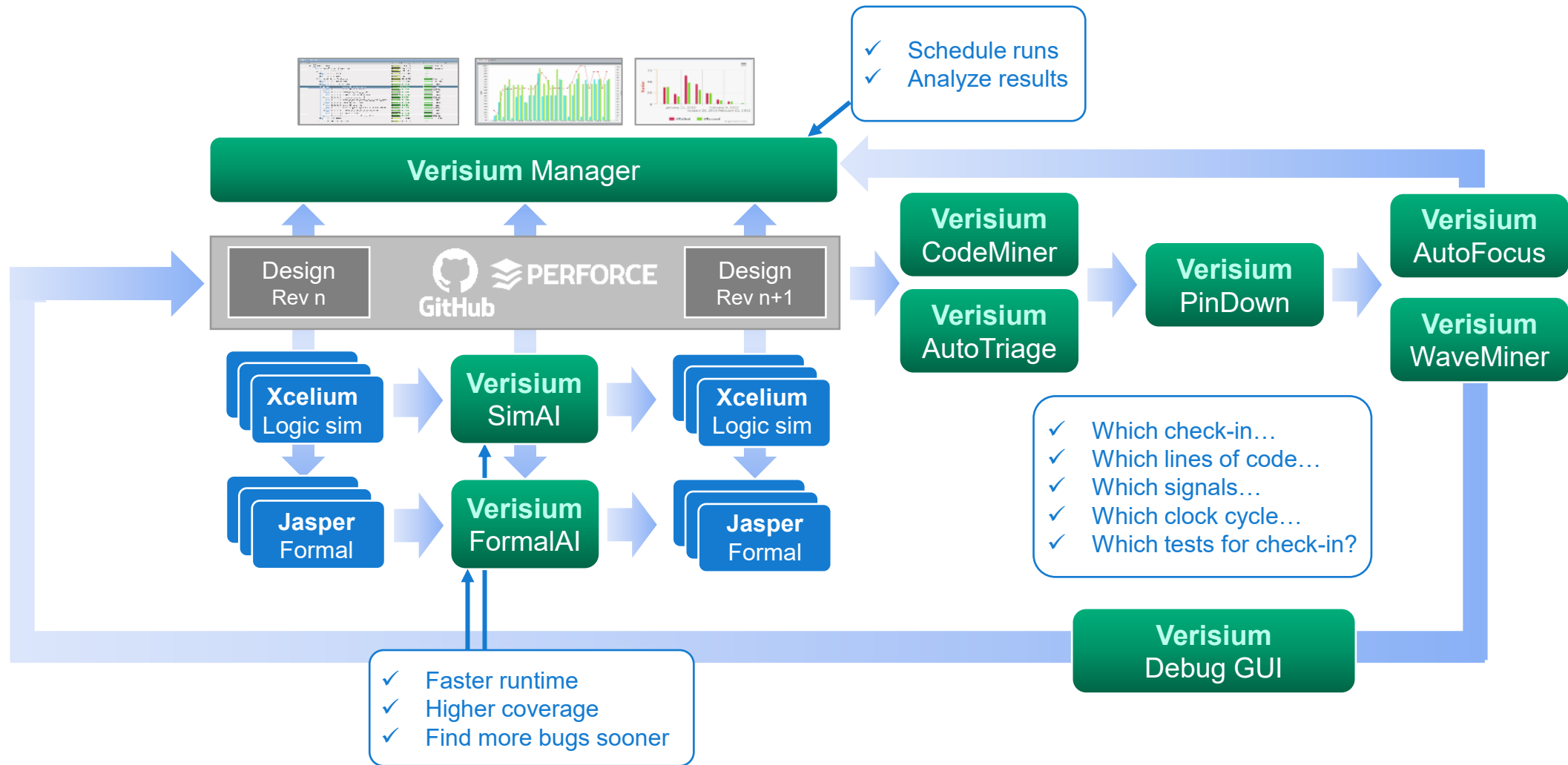


Reinforcement Learning

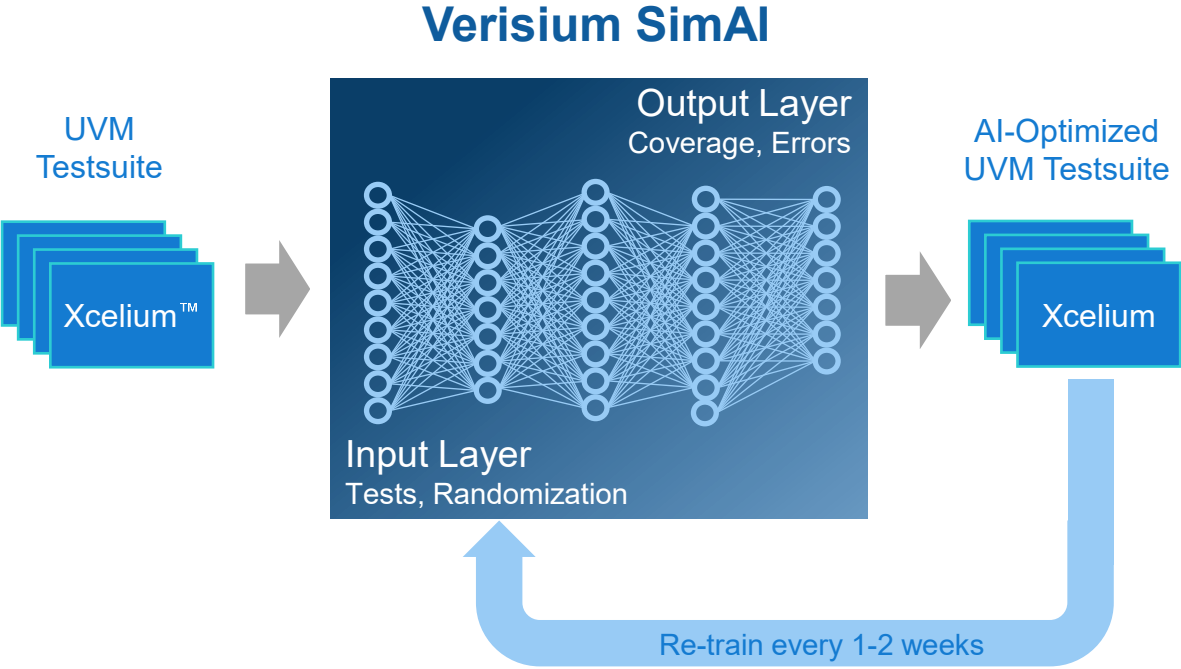
Design	Proof
Block A	1.2x
Block B	2.1x
Block C	1.9x
Block D	1.6x
Block E	1.5x
Block F	2.5x
Block G	3.1x
Block H	1.6x
Block I	3.6x

Optimization of core algorithms and engines with ML (AI Inside)
Using tools data to improve the core algorithms of engines and flows.

Verisium in Action



Xcelium SimAI



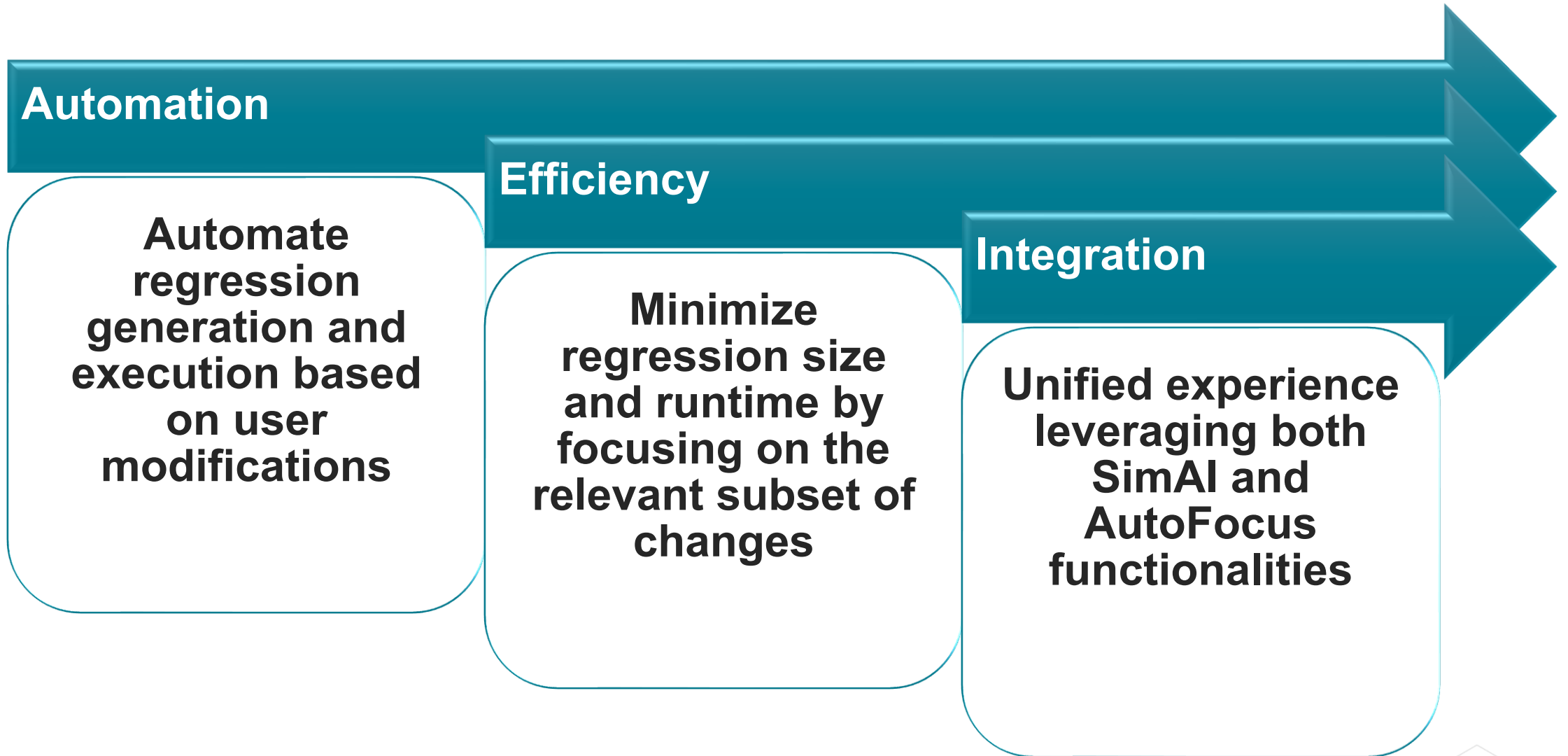
Used by 18 of the top 20 Semis

AI Mode	AI Mode Goal
Compression Mode	<i>Provides compressed test list achieving similar coverage as original full regression</i>
Failure Amp Mode	<i>Amplifies a rare hard to hit failure so it can be predictably reproduced and get fixed quickly.</i>
Coverage Amp Mode	<i>Finds new bugs by amplifying rarely hit on bins</i>
Closure Mode	<i>Hit cover points that have not been hit yet. Works best when run iteratively (run-refine-run-refine...)</i>
AutoFocus Mode*	<i>Automatically generates a customized sanity/check-in gate testuite based on a specific RTL/TB changelist.</i>

* December 2025

Design Type	Speed Up (Compression Mode)
Modem IP	20x
Mobile SoC	7x
AI IP	8x
AI IP	3x
Memory IP	5x
Base Station	18x
NOC IP	5x
Automotive SoC	5x
CPU IP	3x
Network IP	4x
CPU IP	9x

Verisium AI Focused Regression Flow



Regression Health Dashboard

Verisium Manager

RegressionAnalysisPlanningComposer

My_Sessions*

LaunchImportCollect RunsRefreshScripts ManagerExportExport MergeStopStop Auto.SuspendResumeSet as completedDeleteRelocateOpen dirSession InfoRecalc UDAChartFilter SelectedFilter all at once

ViewsGlobal OperationsScriptsSessionsList Tabssession_AB_sim_args.shuri.25_10_06_01_18_20_34AttributesAnalyzePlanningReportHelp

Flow Sessions

Session StatusNameTotal Runs#Passed

(no filter)(no filter)(no filter)(n

completedsession_AB_sim_args.shuri.25_10_06_01_2...324144

completedsession_AB_sim_args.shuri.25_10_06_01_2...22498

completedsession_AB_sim_args.shuri.25_10_06_01_2...224104

completedsession_AB_sim_args.shuri.25_10_06_01_2...22496

completedsession_AB_sim_args.shuri.25_10_06_01_1...22485

completedsession_AB_sim_args.shuri.25_10_06_01_1...224103

completedsession_AB_sim_args.shuri.25_10_06_01_1...22087

completedsession_AB_sim_args.shuri.25_10_06_01_1...239

completedsession_AB_sim_args.shuri.25_10_06_01_0...2315

in_progresscrc_demo.shuri.25_10_06_01_03_26_6453970

in_progresscrc_demo.shuri.25_10_06_00_55_38_5076970

completedcrc_demo.shuri.25_10_06_00_46_58_70576060

completedcrc_demo.shuri.25_10_06_00_41_23_94546060

completedcrc_demo.shuri.25_09_29_22_05_28_74246060

completedcrc_demo.shuri.25_09_29_21_57_02_70156060

completedcrc_demo.shuri.25_09_29_20_18_24_65706060

completedsession_AB_sim_args.shuri.25_09_29_20_0...166

completedsession_AB.shuri.25_09_29_18_30_51_1174135

completedsession_AB_sim_args.shuri.25_09_29_18_3...168

completedsession_AB.shuri.25_09_29_18_24_21_4483135

completedsession_AB_sim_args.shuri.25_09_29_18_2...168

completedsession_AB.shuri.25_09_29_18_16_05_3494135

Showing 22 out of 22 items, 1 item selected

Health dashboardErrorsWarnings/Info

Run Anomalies

Duration (sec.)CPU Time (ms.)Max Memory Usage (MB)Average Memory Usage (MB)Max Runs In Parallel

17054001850001850039

200Max Requested188Max Actual

Actual Index	Index	Name	CPU Time (ms.)	Estimated CPU Time (ms.)	Duration (sec.)	Estimated Duration (sec.)	Max Memory Usage (MB)	Estimated Max Memory Usage (MB)	Average Memory Usage (MB)
(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	(no filter)
2	1	/group_A/A1_test	3689	2759	11	3	237	215	127.0
3	2	/group_A/A1_test	3541	2759	11	3	205	215	137.0
4	3	/group_A/A1_test	3684	2759	12	3	202	215	121.0
5	4	/group_A/A1_test	3772	2759	13	3	228	215	137.0
6	5	/group_A/A1_test	3690	2759	12	3	202	215	124.0
7	6	/group_A/A1_test	3786	2759	12	3	250	215	144.0
8	7	/group_A/A1_test	3625	2759	14	3	270	215	137.0
9	8	/group_A/A1_test	3671	2759	12	3	229	215	139.0

Showing 224 items, 1 item selected

cadenceRegression Center

Messages

Optimization AI in Jasper

Parallel runs find more bugs and achieve deeper proofs



Jasper Smart Proof AI-Driven Orchestration

Proof Profiling Data

- Keep engine-level settings that worked before

Proof Caching

- Reuse the existing result if the constraints and COI are unchanged

Trace Replay

- Leverage previously found traces

Proof Orchestration

- Use Machine Learning to find the best proof algorithm settings

Multi-run optimization

+AI



Find more bugs



Better convergence



Faster proofs

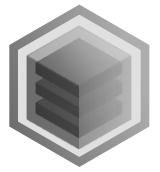
Qualcomm CPU JUG 2023 Best Paper



For fully-converging testcases:

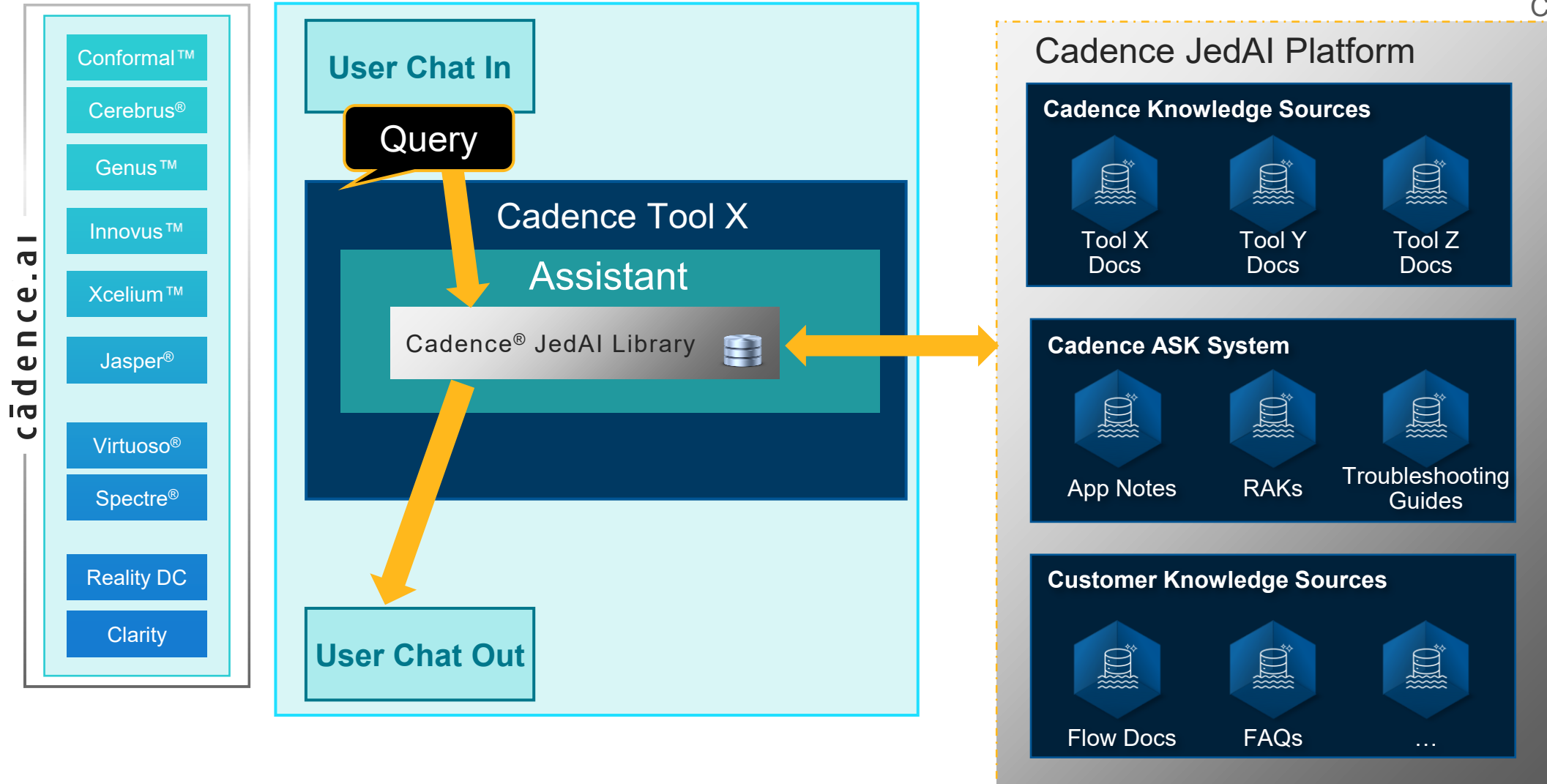
- 5x reduction in runtime
- 90x reduction in machine slots

Conversational LLM Agents Across the Software Portfolio



LEVEL 2

Conversational LLM

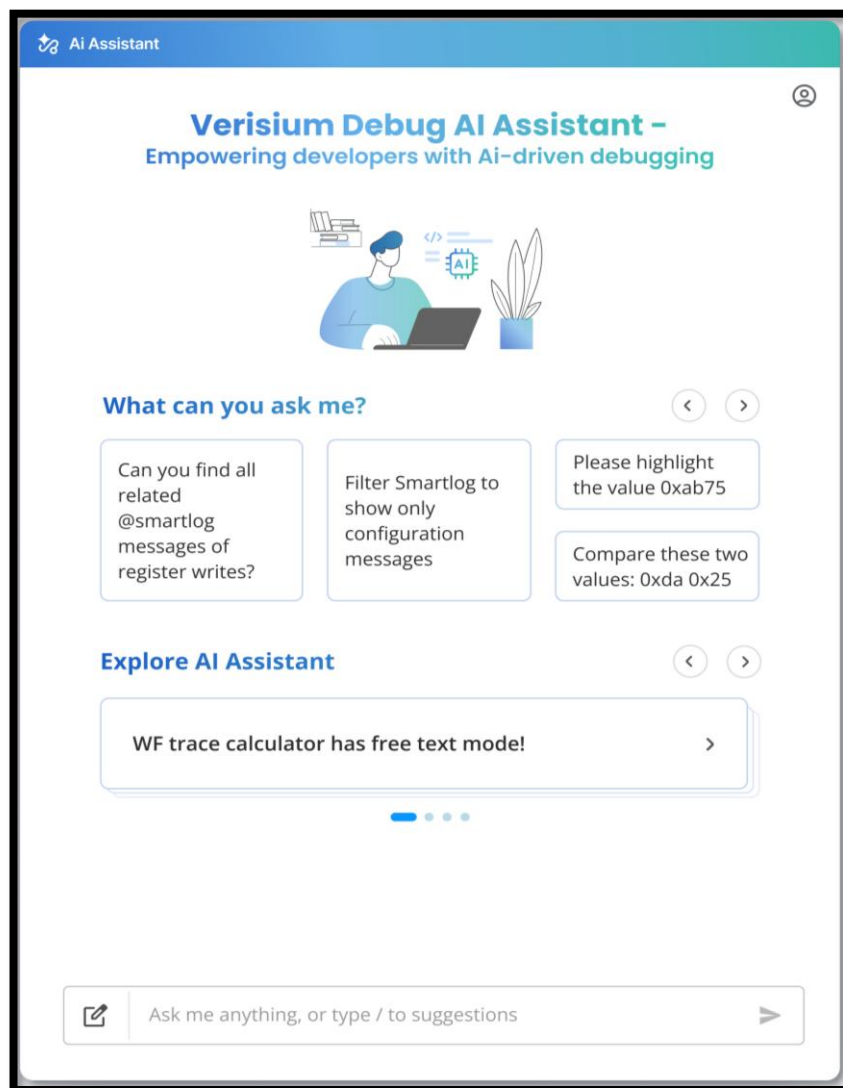


AI Assistants - Verisium Debug AI Assistant



LEVEL 2

Conversational LLM



AI Chat

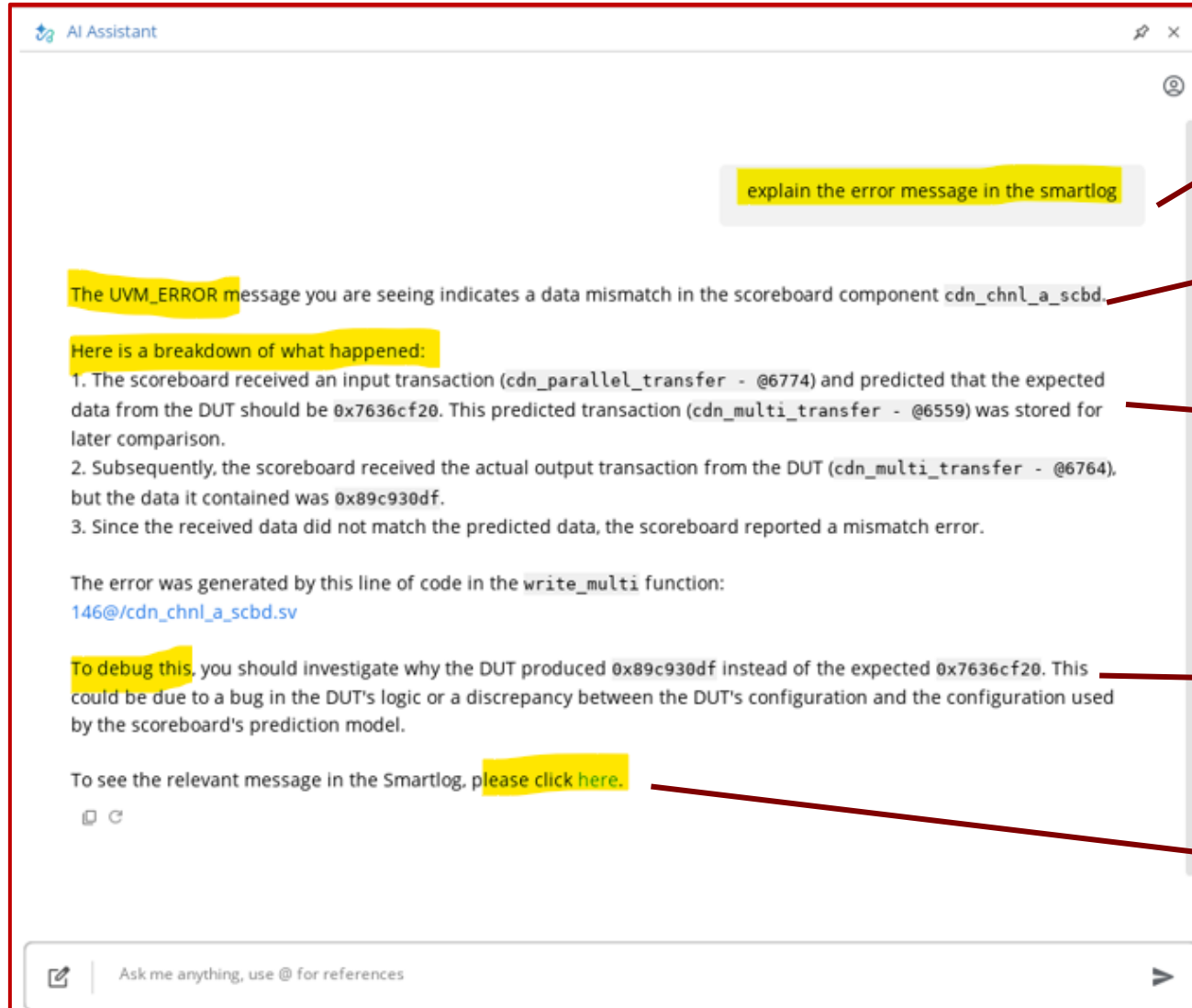
- **Ask and Retrieve** information from your design (Design, TB)
- **Operate** Verisium Debug and summarizing information
- **Answers** from Tool help and Setting options

AI-Powered Tools

- Code inspection - Read code and suggest potential issues – **Autonomous**
- Explain signal - collect data about signal and summarize it's functionality
- Complex waveform expression
- Randomization contradiction AI summary



AI Assistants - Verisium Debug AI Assistant -*SmartLog*, Source Search



Prompt

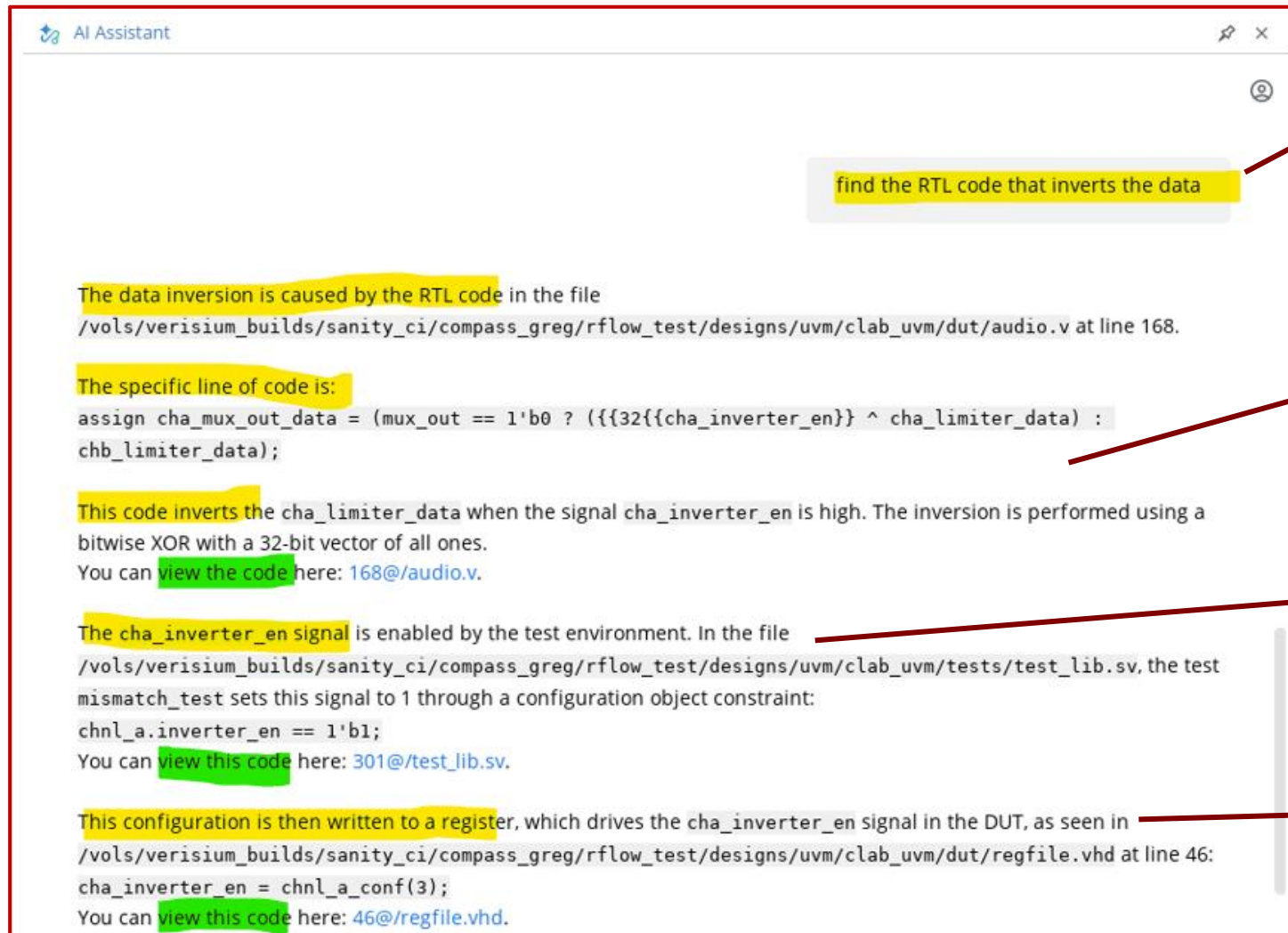
Explain the error message

Explain what leads to the error

Suggest next steps

Links to related SmartLog messages

AI Assistants - Verisium Debug AI Assistant – *RTL exploration*



The screenshot shows a chat window titled "AI Assistant". The prompt is "find the RTL code that inverts the data". The response is as follows:

The data inversion is caused by the RTL code in the file
`/vols/verisium_builds/sanity_ci/compass_greg/rflow_test/designs/uvm/clab_uvm/dut/audio.v` at line 168.

The specific line of code is:

```
assign cha_mux_out_data = (mux_out == 1'b0 ? ({32{cha_inverter_en}} ^ cha_limiter_data) :  
chb_limiter_data);
```

This code inverts the `cha_limiter_data` when the signal `cha_inverter_en` is high. The inversion is performed using a bitwise XOR with a 32-bit vector of all ones.
You can view the code here: [168@/audio.v](#).

The `cha_inverter_en` signal is enabled by the test environment. In the file
`/vols/verisium_builds/sanity_ci/compass_greg/rflow_test/designs/uvm/clab_uvm/tests/test_lib.sv`, the test `mismatch_test` sets this signal to 1 through a configuration object constraint:

```
chnl_a.inverter_en == 1'b1;
```

You can view this code here: [301@/test_lib.sv](#).

This configuration is then written to a register, which drives the `cha_inverter_en` signal in the DUT, as seen in
`/vols/verisium_builds/sanity_ci/compass_greg/rflow_test/designs/uvm/clab_uvm/dut/regfile.vhd` at line 46:

```
cha_inverter_en = chnl_a_conf(3);
```

You can view this code here: [46@/regfile.vhd](#).

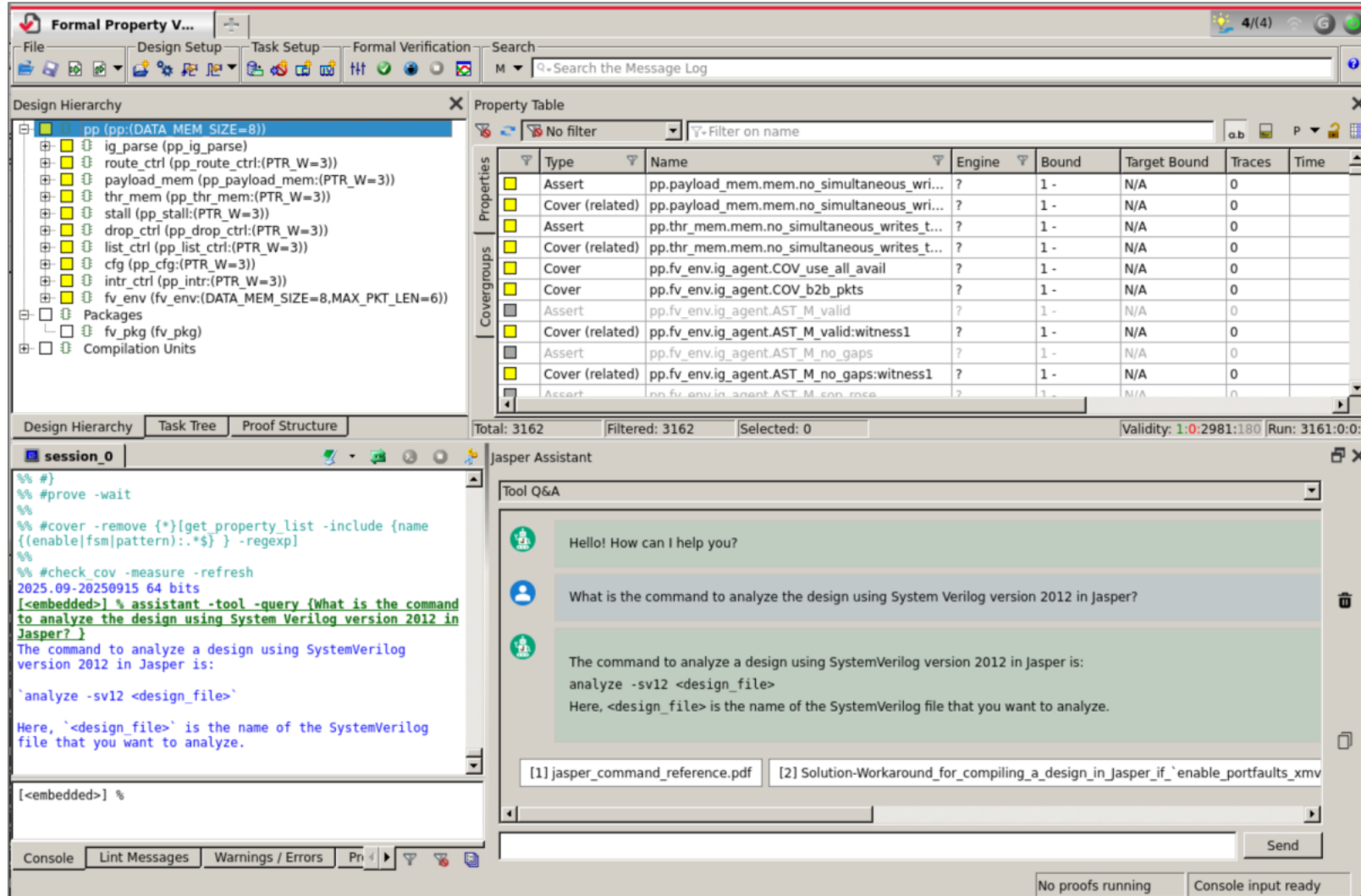
Prompt

Found the RTL logic, explain the logic and provide links

Details about the TB code that enabled the inversion

Details on the Register that enables the inversion

AI Assistants - Jasper AI Assistant – Tool Docs Q&A



The screenshot displays the Formal Property Verifier (FPV) interface. The top section shows the Design Hierarchy with a tree view of components like `pp (pp:(DATA MEM SIZE=8))`, `ig_parse (pp_ig_parse)`, `route_ctrl (pp_route_ctrl:(PTR_W=3))`, `payload_mem (pp_payload_mem:(PTR_W=3))`, `thr_mem (pp_thr_mem:(PTR_W=3))`, `stall (pp_stall:(PTR_W=3))`, `drop_ctrl (pp_drop_ctrl:(PTR_W=3))`, `list_ctrl (pp_list_ctrl:(PTR_W=3))`, `cfg (pp_cfg:(PTR_W=3))`, `intr_ctrl (pp_intr:(PTR_W=3))`, and `fv_env (fv_env:(DATA MEM SIZE=8,MAX_PKT_LEN=6))`. The middle section shows the Property Table with columns for Type, Name, Engine, Bound, Target Bound, Traces, and Time. The bottom section shows the Jasper Assistant chat window with a session log and a chat interface.

Design Hierarchy

- pp (pp:(DATA MEM SIZE=8))
 - ig_parse (pp_ig_parse)
 - route_ctrl (pp_route_ctrl:(PTR_W=3))
 - payload_mem (pp_payload_mem:(PTR_W=3))
 - thr_mem (pp_thr_mem:(PTR_W=3))
 - stall (pp_stall:(PTR_W=3))
 - drop_ctrl (pp_drop_ctrl:(PTR_W=3))
 - list_ctrl (pp_list_ctrl:(PTR_W=3))
 - cfg (pp_cfg:(PTR_W=3))
 - intr_ctrl (pp_intr:(PTR_W=3))
 - fv_env (fv_env:(DATA MEM SIZE=8,MAX_PKT_LEN=6))
- Packages
 - fv_pkg (fv_pkg)
- Compilation Units

Property Table

Type	Name	Engine	Bound	Target Bound	Traces	Time
Assert	pp.payload_mem.mem.no_simultaneous_wri...	?	1 -	N/A	0	
Cover (related)	pp.payload_mem.mem.no_simultaneous_wri...	?	1 -	N/A	0	
Assert	pp.thr_mem.mem.no_simultaneous_writes_t...	?	1 -	N/A	0	
Cover (related)	pp.thr_mem.mem.no_simultaneous_writes_t...	?	1 -	N/A	0	
Cover	pp.fv_env.ig_agent.COV_use_all_avail	?	1 -	N/A	0	
Cover	pp.fv_env.ig_agent.COV_b2b_pkts	?	1 -	N/A	0	
Assert	pp.fv_env.ig_agent.AST_M_valid	?	1 -	N/A	0	
Cover (related)	pp.fv_env.ig_agent.AST_M_valid:witness1	?	1 -	N/A	0	
Assert	pp.fv_env.ig_agent.AST_M_no_gaps	?	1 -	N/A	0	
Cover (related)	pp.fv_env.ig_agent.AST_M_no_gaps:witness1	?	1 -	N/A	0	
Assert	pp.fv_env.ig_agent.AST_M_no_race	?	1 -	N/A	0	

Jasper Assistant

session_0

```
%% #)
%% #prove -wait
%%
%% #cover -remove {*}[get_property_list -include {name
%% {(enable|fsm|pattern):.*}] } -regexp]
%%
%% #check_cov -measure -refresh
2025.09-20250915 64 bits
[embedded]> % assistant -tool -query {What is the command
to analyze the design using System Verilog version 2012 in
Jasper? }
The command to analyze a design using SystemVerilog
version 2012 in Jasper is:
'analyze -sv12 <design_file>'
Here, '<design_file>' is the name of the SystemVerilog
file that you want to analyze.

[embedded]> %
```

Tool Q&A

Hello! How can I help you?

What is the command to analyze the design using System Verilog version 2012 in Jasper?

The command to analyze a design using SystemVerilog version 2012 in Jasper is:
analyze -sv12 <design_file>
Here, <design_file> is the name of the SystemVerilog file that you want to analyze.

[1] jasper_command_reference.pdf [2] Solution-Workaround_for_compiling_a_design_in_Jasper_if_enable_portfaults_xmrv

Send

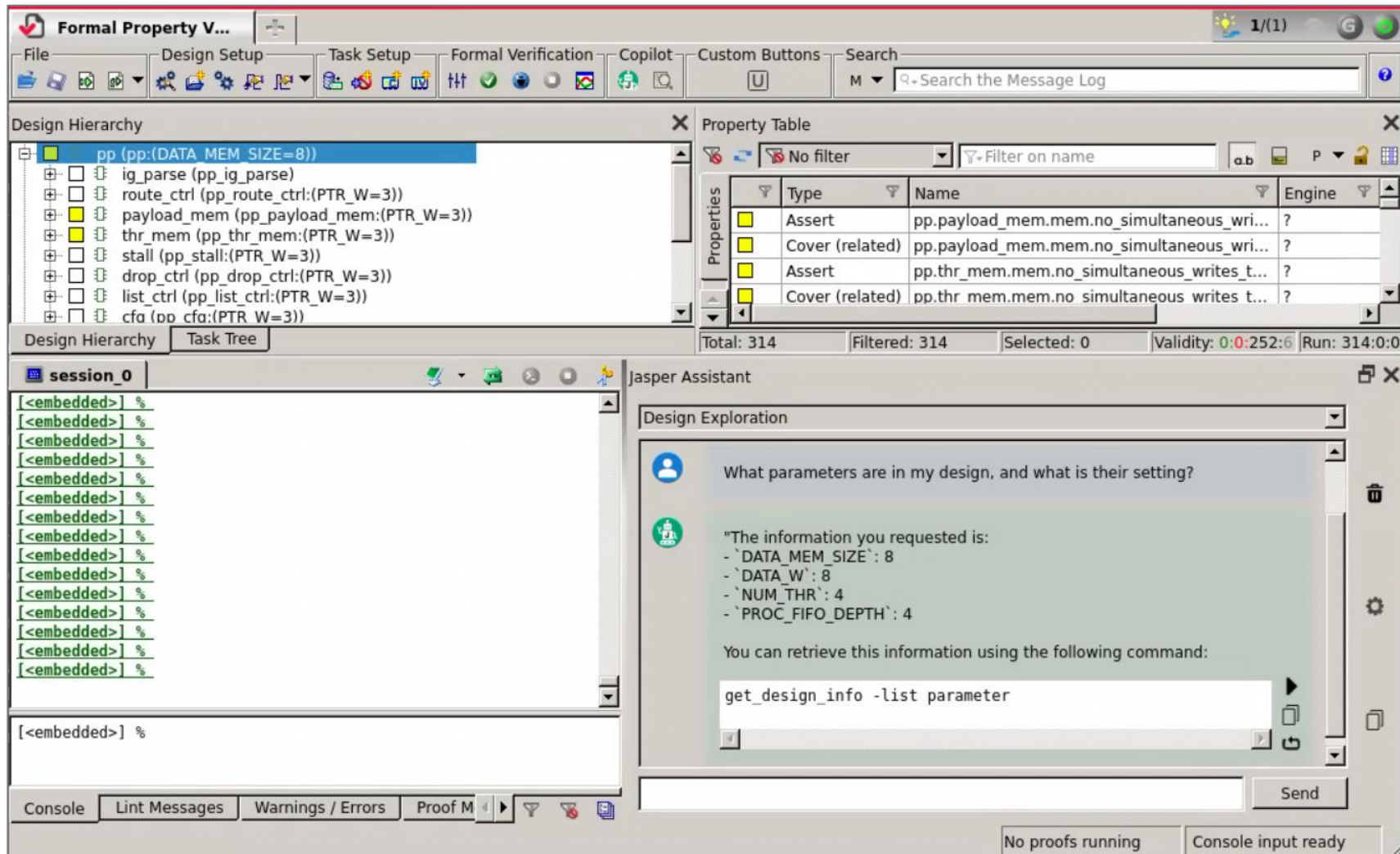
No proofs running Console input ready

- ✓ **Tool documentation search** and command querying from NL
- ✓ Connection to Cadence or customer-provided LLM via JedAI
- ✓ Open queries to streamline general routines
- ✓ History

Query examples

- **How to convert** the assertion `pp.fv_env.fl_chk.AST_no_duplicate` to an assume?
- Give me an example of **commands** to run hunt cycle swarm
- How to save the DHO database
- What is target bound?
- How to **enable multiple traces** for an assertion

AI Assistants - Jasper AI Assistant – Design Exploration



- ✓ **Design exploration** from natural language
- ✓ Tcl code editable in case of required adjustments

Query examples

- **List all IOs** of instance <instance>
- What are the **instances** with parameter PTR_W set to 3?
- Which assertions are disabled?
- What are the **FIFOs** of instance <instance>?
- What are the instances of module <module>?

AI Assistants - Jasper AI Assistant – Tcl Property Generation

The screenshot displays the Jasper tool interface, which is used for formal verification. The main window is titled "Formal Property V...".

Design Hierarchy: The left pane shows a hierarchical view of the design. The root is "pp (pp:(DATA_MEM_SIZE=8))". It contains several sub-modules and components, including "ig_parse (pp_ig_parse)", "route_ctrl (pp_route_ctrl:(PTR_W=3))", "payload_mem (pp_payload_mem:(PTR_W=3))", "thr_mem (pp_thr_mem:(PTR_W=3))", "stall (pp_stall:(PTR_W=3))", "drop_ctrl (pp_drop_ctrl:(PTR_W=3))", "list_ctrl (pp_list_ctrl:(PTR_W=3))", "cfg (pp_cfg:(PTR_W=3))", "intr_ctrl (pp_intr:(PTR_W=3))", "fv_env (fv_env:(DATA_MEM_SIZE=8,DYNAMIC_SCB=0,MAX_PKT_LEN=6))", "Packages", "fv_pkg (fv_pkg)", and "Compilation Units".

Property Table: The right pane displays a table of properties. The table has columns: Type, Name, Engine, Bound, and Target. The properties are categorized into "Properties" and "Covergroups".

Type	Name	Engine	Bound	Target
Assert	pp.payload_mem.mem.no_simultaneous_wri...	?	1 -	N/A
Cover (related)	pp.payload_mem.mem.no_simultaneous_wri...	Bm	1	N/A
Assert	pp.thr_mem.mem.no_simultaneous_writes_t...	?	1 -	N/A
Cover (related)	pp.thr_mem.mem.no_simultaneous_writes_t...	Bm	1	N/A
Cover	pp.fv_env.ig_agent.COV_use_all_avail	Bm	28	N/A
Cover	pp.fv_env.ig_agent.COV_b2b_pkts	Bm	23	N/A
Assert	pp.fv_env.ig_agent.AST_M_valid	?	1 -	N/A
Cover (related)	pp.fv_env.ig_agent.AST_M_valid:witness1	Ht	21	N/A
Assert	pp.fv_env.ig_agent.AST_M_no_gaps	?	1 -	N/A

Session 0: The bottom-left pane shows the session log. It contains a list of embedded properties and a summary of the property generation process.

Jasper Assistant: The bottom-right pane shows the "Tcl Property Generation" dialog box. It contains a description of the property, a name field, an expression field, and status indicators.

Property Name: eg_req_ptr_nx_not_all_high

Expression: eg_req |-> !&fv_env.fl_chk.ptrs_nx

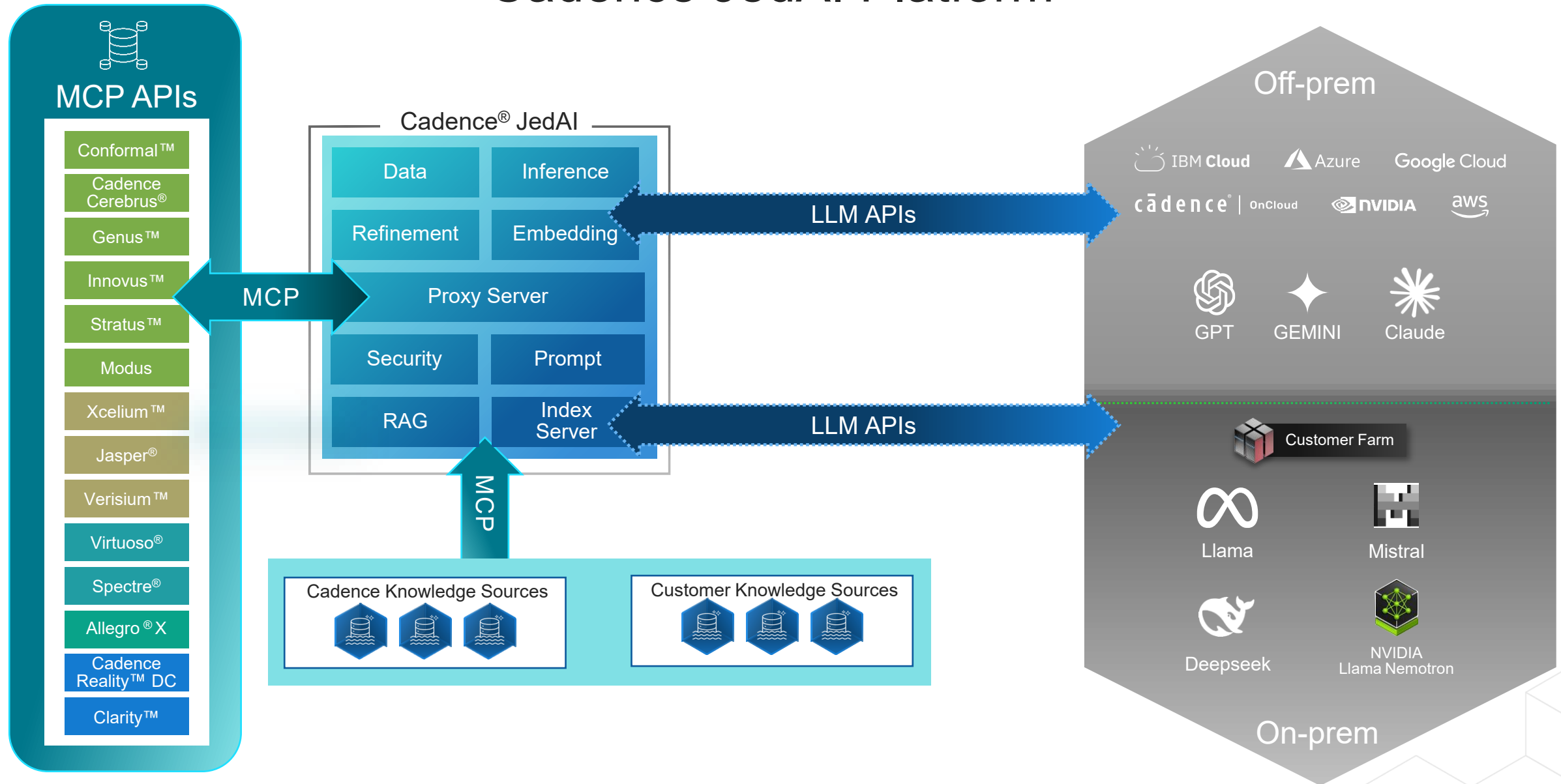
Status: Valid property name, No syntax error

- ✓ Tcl **property generation** from natural language
- ✓ Specialized prompt for SVA generation
- ✓ Instant syntax checking, including signal names
- ✓ Toggle between different property types

Query examples

- **Check that** sig_a is equal to DATA_MEM_SIZE if initialized and all bits of sig_b are high
- **Check that** sig_a is different from DATA_MEM_SIZE if sig_b is high and any bit of sig_c is high, but not all bits
- **Check that** sig_a is always equal or more than one and sig_b is equal or less than four

Cadence JedAI Platform



New Protocols Are Enabling Plug-and-Play Agentic Systems

Anthropic



Model Context Protocol (MCP)

Google

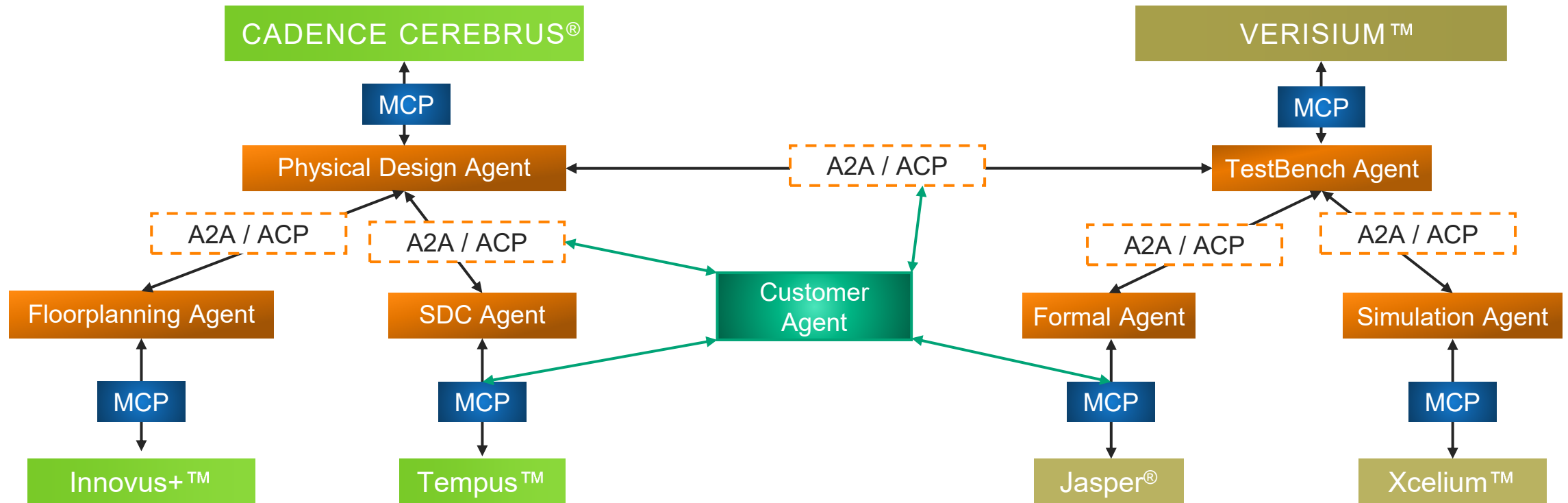
A2A protocol

Agent2Agent (A2A)

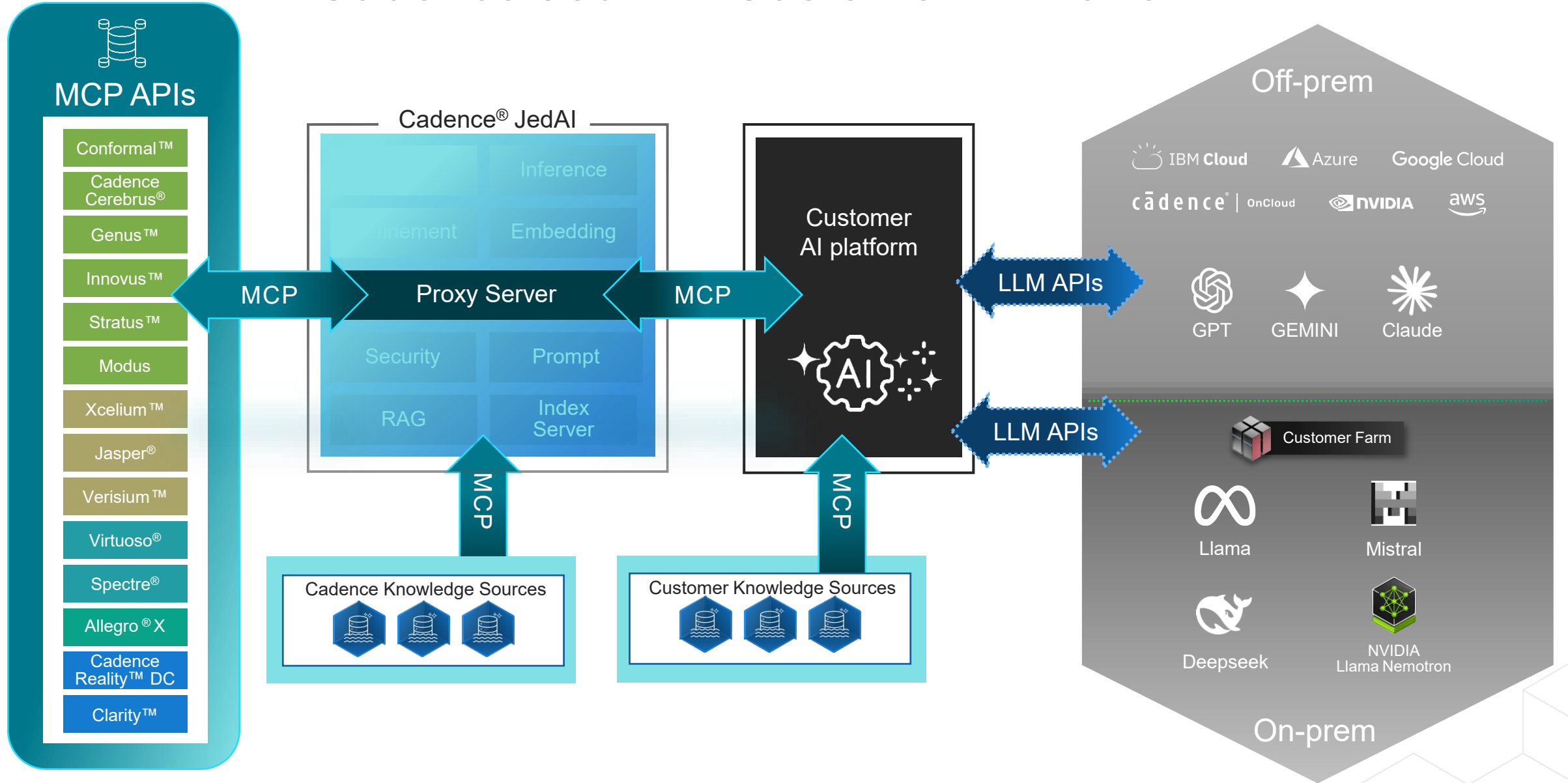
IBM

BeeAI

Agent Communication Protocol (ACP)



Cadence JedAI + Customer AI Platform



Jasper Assistant Script Generation Agent



LEVEL 3
Complex Reasoning

The screenshot shows the Formal Property V... interface. The Design Hierarchy on the left lists components like `pp (pp:(DATA_MEM_SIZE=8))` and `ig_parse (pp_ig_parse)`. The Property Table on the right lists properties such as `pp.payload_mem.mem.no_simultaneous_wri...` and `pp.thr_mem.mem.no_simultaneous_writes_t...`. The chat window on the right shows a conversation with the Jasper Assistant, including a greeting and a request to create a Tcl proc. The console at the bottom displays the output of the `% prove_liveness_assertions 30 3 8` command, showing settings for the proof thread.

Type	Name	Engine	Bound	Traces	Tir
Assert	pp.payload_mem.mem.no_simultaneous_wri...	?	1 -	0	
Cover (related)	pp.payload_mem.mem.no_simultaneous_wri...	?	1 -	0	
Assert	pp.thr_mem.mem.no_simultaneous_writes_t...	?	1 -	0	
Cover (related)	pp.thr_mem.mem.no_simultaneous_writes_t...	?	1 -	0	
Cover	pp.fv_env.ig_agent.COV_use_all_avail	?	1 -	0	
Cover	pp.fv_env.ig_agent.COV_b2b_pkts	?	1 -	0	
Assert	pp.fv_env.ig_agent.AST_M_valid	?	1 -	0	

```
assert}}  
  
# Prove the property list with specified parameters  
prove -property $property_list -per_engine_max_jobs $Y -  
max_jobs $Z -time_limit ${X}m  
  
# Report the properties still undetermined  
set undetermined_properties [get_property_list -include {status  
undetermined}]  
return $undetermined_properties  
}  
  
[<embedded>] % prove_liveness_assertions 30 3 8  
INFO (IPF036): Starting proof on task: "<embedded>", 2 properties  
to prove with 0 already proven/unreachable  
INFO (IPF031): Settings used for proof thread 5:  
orchestration = on  
time_limit = 1800s  
per_property_time_limit = 1s * 10 ^ scan  
engine_mode = auto  
proofgrid_per_engine_max_jobs = 3  
proofgrid_max_jobs = 8  
max engine jobs = auto (max 8)  
proofgrid_mode = local  
proofgrid_cloud_file_upload = auto  
proofgrid_restarts = 10  
  
[<embedded>] %
```

Chat window content:

JedAI: Hello! How can I help you?

User: Create a Tcl proc that proves all liveness assertions in my task for X minutes, using Y per engine max jobs and Z max jobs. After the proof is done, the proc should report the properties still undetermined.

JedAI: `proc prove_liveness_assertions {X Y Z} {
 # List all liveness assertions in the current task
 set property_list [get_property_list -include {liveness 1 type assert}]

 # Prove the property list with specified parameters
 prove -property $property_list -per_engine_max_jobs $Y -max_jobs $Z -t

 # Report the properties still undetermined
 set undetermined_properties [get_property_list -include {status unde
 return $undetermined_properties
}`

- ✓ Jasper Tcl script generation from NL
- ✓ Under the hood reasoning about command formation
- ✓ Edit return
- ✓ Execute on the fly
- ✓ History

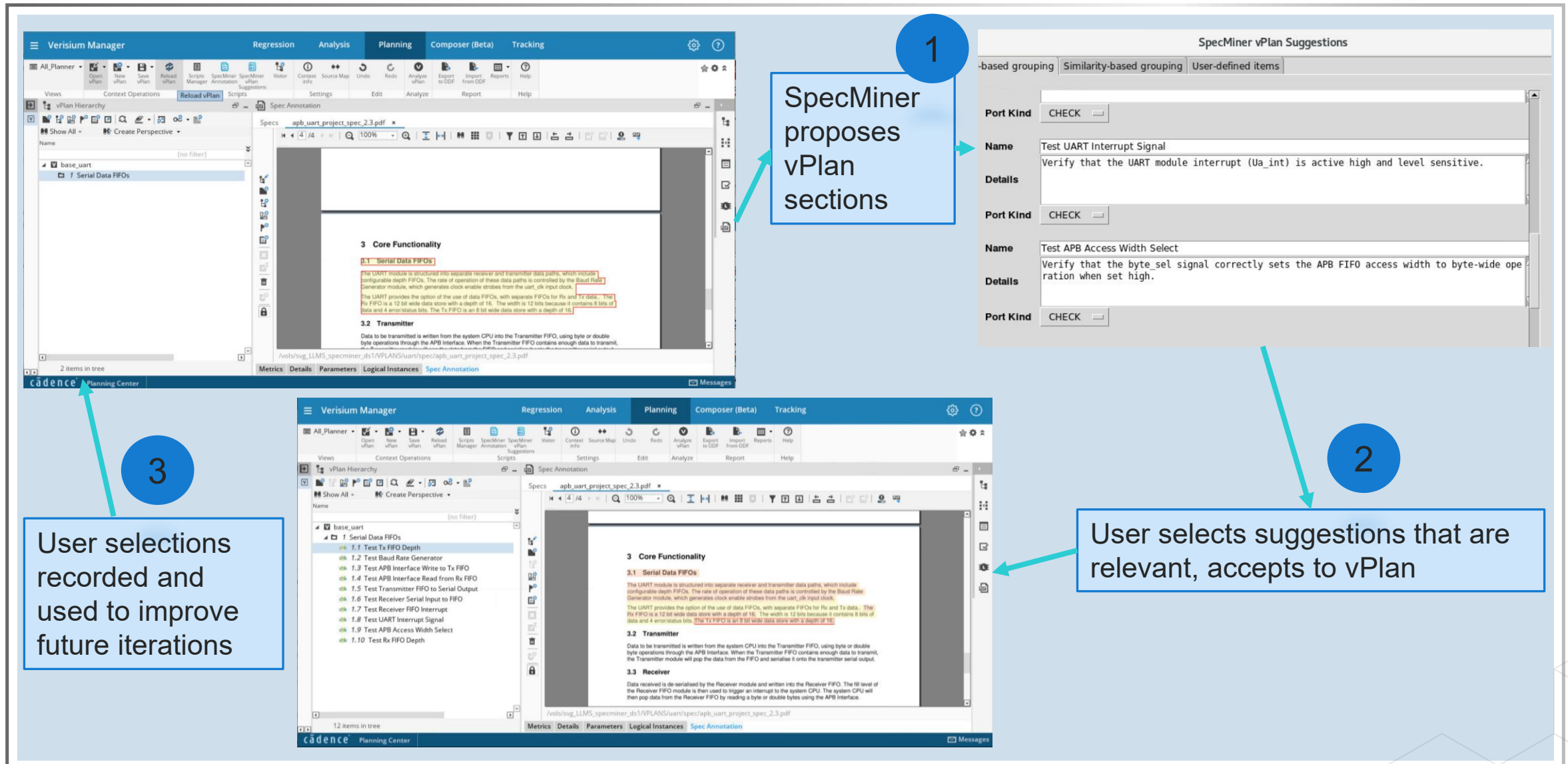
Reasoning

Edit

Execute

This slide contains forward-looking statements regarding Cadence's business or products. Actual results may differ materially from the information presented here.

Agentic AI in Verisium – Verification Plan Gen



1 SpecMiner proposes vPlan sections

2 User selects suggestions that are relevant, accepts to vPlan

3 User selections recorded and used to improve future iterations

The diagram illustrates the workflow of SpecMiner vPlan Suggestions in Verisium Manager. It shows the user interface with the vPlan Hierarchy, Spec Annotation, and SpecMiner vPlan Suggestions panels. The workflow is numbered 1 through 3, indicating the sequence of actions from proposal to selection and recording.

SpecMiner vPlan Suggestions

- Port Kind: CHECK
- Name: Test UART Interrupt Signal
- Details: Verify that the UART module interrupt (Ua_int) is active high and level sensitive.
- Port Kind: CHECK
- Name: Test APB Access Width Select
- Details: Verify that the byte_sel signal correctly sets the APB FIFO access width to byte-wide operation when set high.
- Port Kind: CHECK

Verisium Manager

Regression Analysis Planning Composer (Beta) Tracking

Views: Context Operations Reload vPlan Scripts SpecMiner vPlan Annotations vPlan Suggestions

Spec Annotation

Specs: apb_uart_project_spec_2.3.pdf

3 Core Functionality

3.1 Serial Data FIFOs

The UART module is structured into separate receiver and transmitter data paths, which include configurable depth FIFOs. The rate of operation of these data paths is controlled by the Baud Rate Generator module, which generates clock enable strobes from the uart_clk input clock.

The UART provides the option of the use of data FIFOs, with separate FIFOs for Rx and Tx data. The Rx FIFO is a 12 bit wide data store with a depth of 16. The width is 12 bits because it contains 8 bits of data and 4 error/status bits. The Tx FIFO is an 8 bit wide data store with a depth of 16.

3.2 Transmitter

Data to be transmitted is written from the system CPU into the Transmitter FIFO, using byte or double byte operations through the APB interface. When the Transmitter FIFO contains enough data to transmit, the Transmitter module will pop the data from the FIFO and serialise it onto the transmitter serial output.

3.3 Receiver

Data received is deserialised by the Receiver module and written into the Receiver FIFO. The 16 level of the Receiver FIFO module is then used to trigger an interrupt to the system CPU. The system CPU will then pop data from the Receiver FIFO by reading a byte or double bytes using the APB interface.

2 Items in tree

cadence Planning Center

Verisium Manager

Regression Analysis Planning Composer (Beta) Tracking

Views: Context Operations Reload vPlan Scripts SpecMiner vPlan Annotations vPlan Suggestions

Spec Annotation

Specs: apb_uart_project_spec_2.3.pdf

3 Core Functionality

3.1 Serial Data FIFOs

The UART module is structured into separate receiver and transmitter data paths, which include configurable depth FIFOs. The rate of operation of these data paths is controlled by the Baud Rate Generator module, which generates clock enable strobes from the uart_clk input clock.

The UART provides the option of the use of data FIFOs, with separate FIFOs for Rx and Tx data. The Rx FIFO is a 12 bit wide data store with a depth of 16. The width is 12 bits because it contains 8 bits of data and 4 error/status bits. The Tx FIFO is an 8 bit wide data store with a depth of 16.

3.2 Transmitter

Data to be transmitted is written from the system CPU into the Transmitter FIFO, using byte or double byte operations through the APB interface. When the Transmitter FIFO contains enough data to transmit, the Transmitter module will pop the data from the FIFO and serialise it onto the transmitter serial output.

3.3 Receiver

Data received is deserialised by the Receiver module and written into the Receiver FIFO. The 16 level of the Receiver FIFO module is then used to trigger an interrupt to the system CPU. The system CPU will then pop data from the Receiver FIFO by reading a byte or double bytes using the APB interface.

12 Items in tree

cadence Planning Center

Agentic Workflow Example

vManager AI Assistant

JedAI Platform

Name	Run Script	Stat
(no filter)	(no filter)	[fa
/regression_tests/apb_uart_rx_tx	TESTNAME=apb_uart_rx_tx	✖
/regression_tests/apb_uart_rx_tx	TESTNAME=apb_uart_rx_tx	✖
/regression_tests/error_reg_seq_test	NAME=error_reg_seq_test	✖

vManager AI Assistant – group failures and launch agents to root cause and fix.

vDebug AI Assistant

JedAI Platform

```

h1
wr_ptr <= #1 'h0;
'h1
rd_ptr <= #1
end // end if
else begin
if(wr_en & (~full))
'h0
'h1
wr_ptr <= #1 wr_ptr+1;
end

```

Inspect Code For Potential Issues
Copy Text
Copy File Path
Syntax

vDebug AI Assistant - root-cause the failure and generate summary of potential issues.

Frontend Silicon Agent

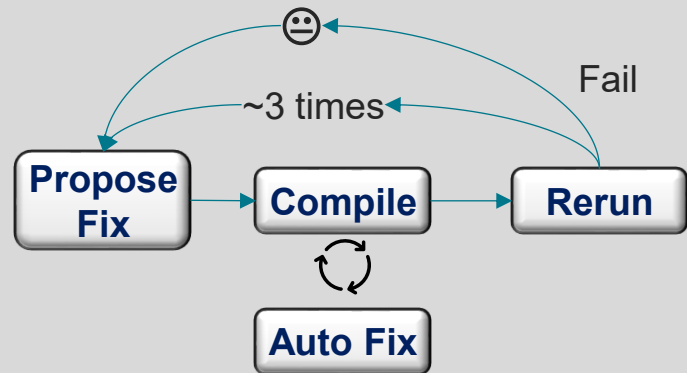
JedAI Platform

```

radio: radio@40001000 {
  bt_hci_sdc: bt_hci_sdc {
    status = "okay";
  };
  bt_hci_controller: bt_hci_controller {
    compatible = "zephyr,bt-hci-ll-sw-split";
    status = "disabled";
  };
};

```

Frontend Silicon Agent – receives the issue description, proposes fix, applies fix, and loops until it is resolved.

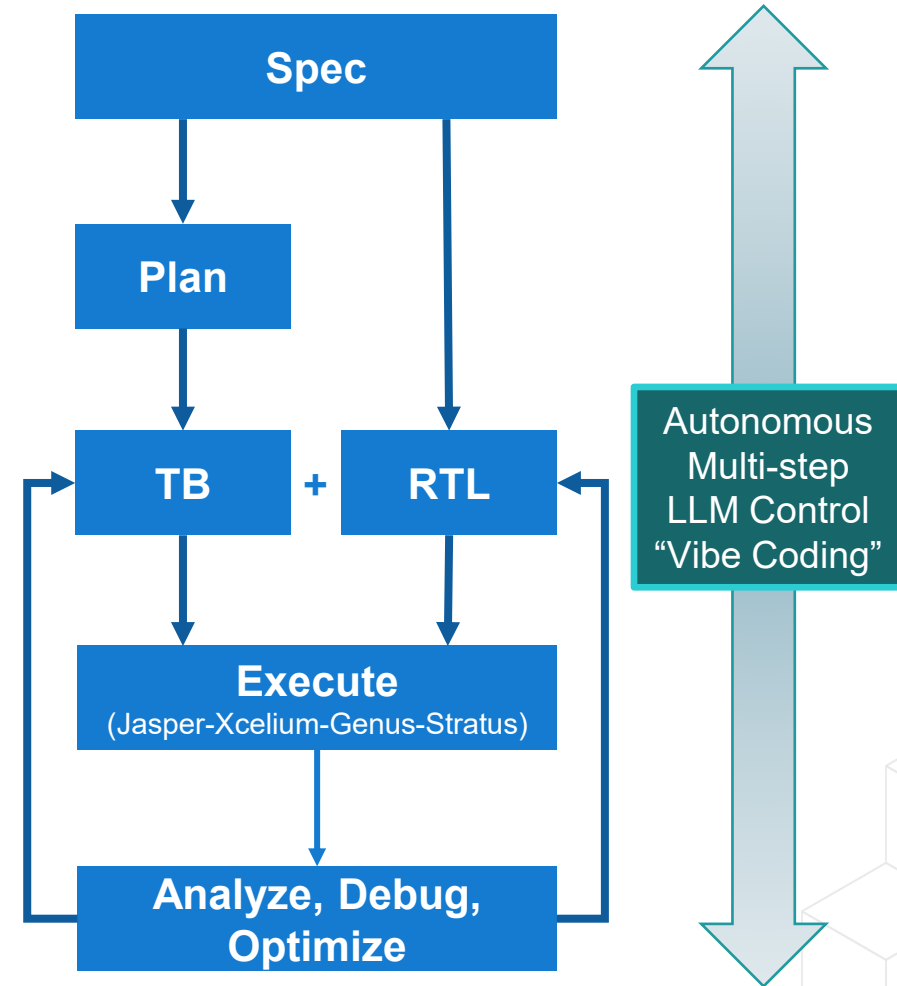
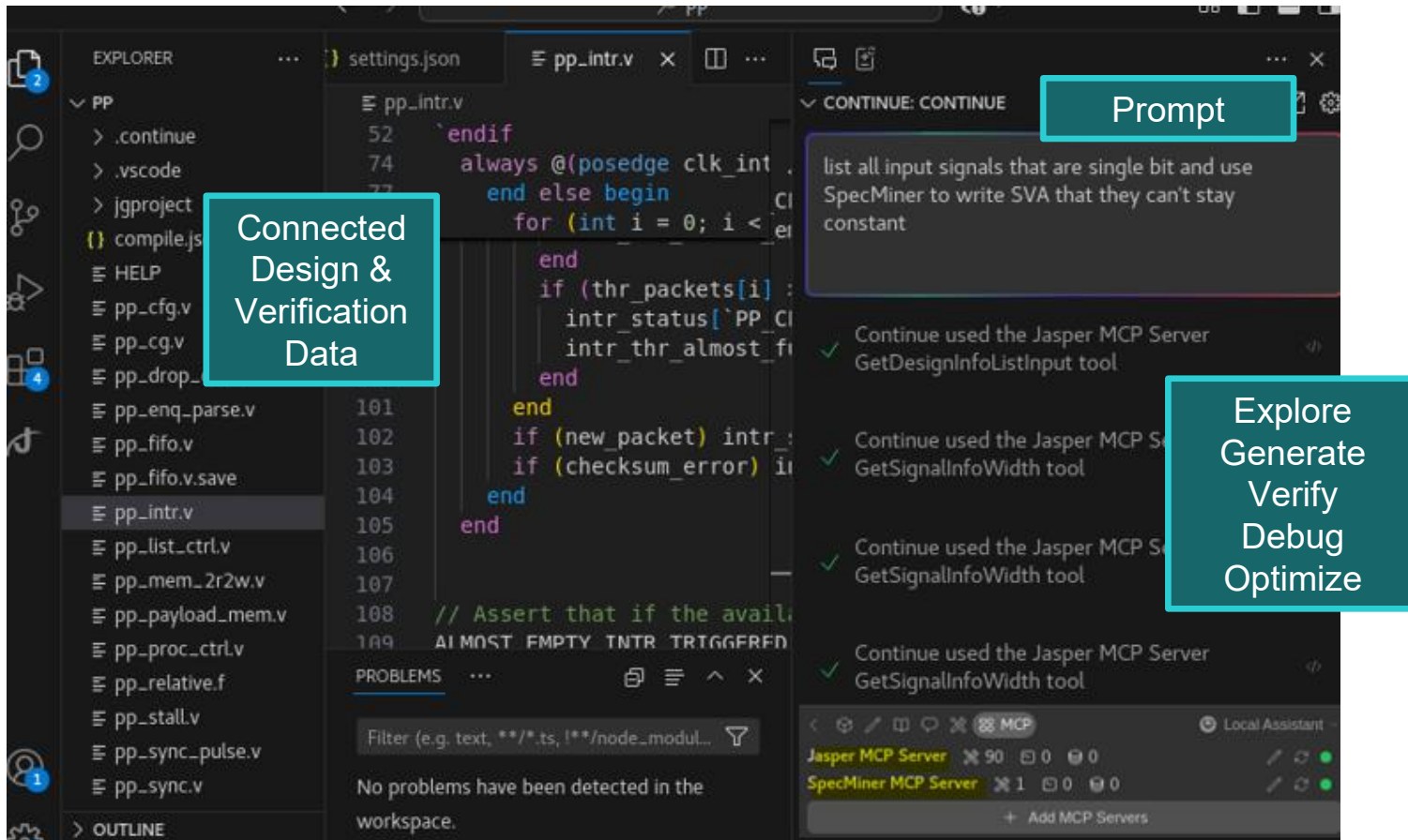


Cadence Frontend Agent for RTL and Testbench Creation



VS Code Extension

JedAI Platform



Cadence Welcome's Chipstack

The World's First AI Verification Engineer

Business Impact

10x Faster

Testbench and coverage generation

60-80% Automation

Eliminates manual DV tasks

Production Ready

Deployed at leading customers
battle tested at scale

Enterprise Security

On-prem or cloud,
your IP stays secure



ChipStack AI Platform

Design
Understanding

Formal
Verification

Unit
Testing

UVM
Generation

AI Chat
Assistant

Coverage
Analysis

Agentic • Integrated • Automated

✓ **FULL BATCH MODE CLI**

Cadence Integration

Native tool integration



Xcelium

Execution Manager, VIPs



Jasper Formal

Property verification



vDebug

Debug and root cause



vManager

Verification management



VWDB / UCM/UCD

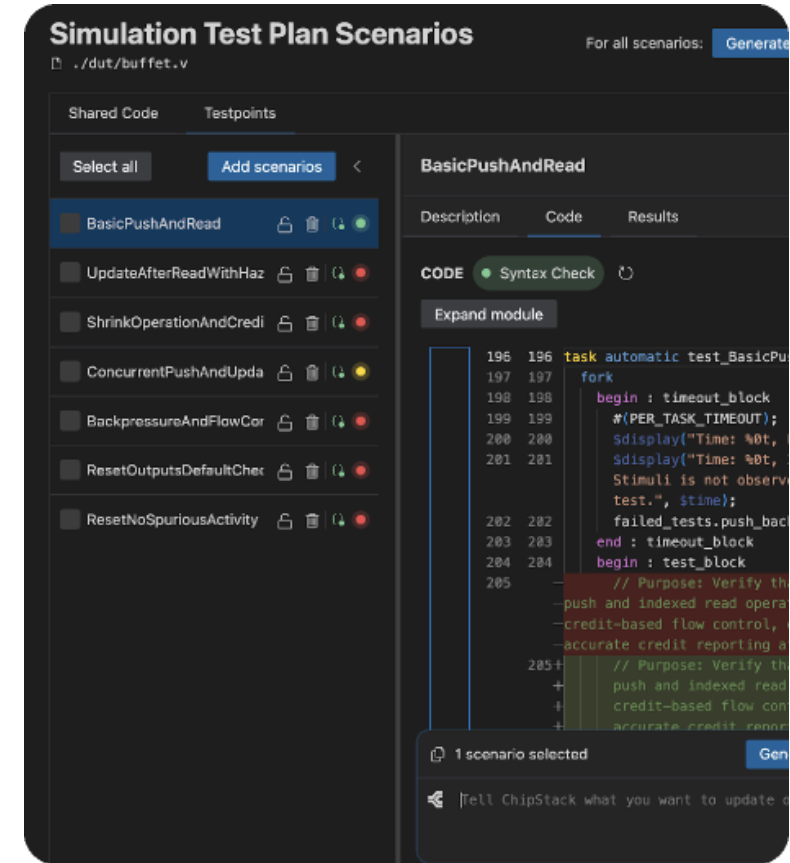
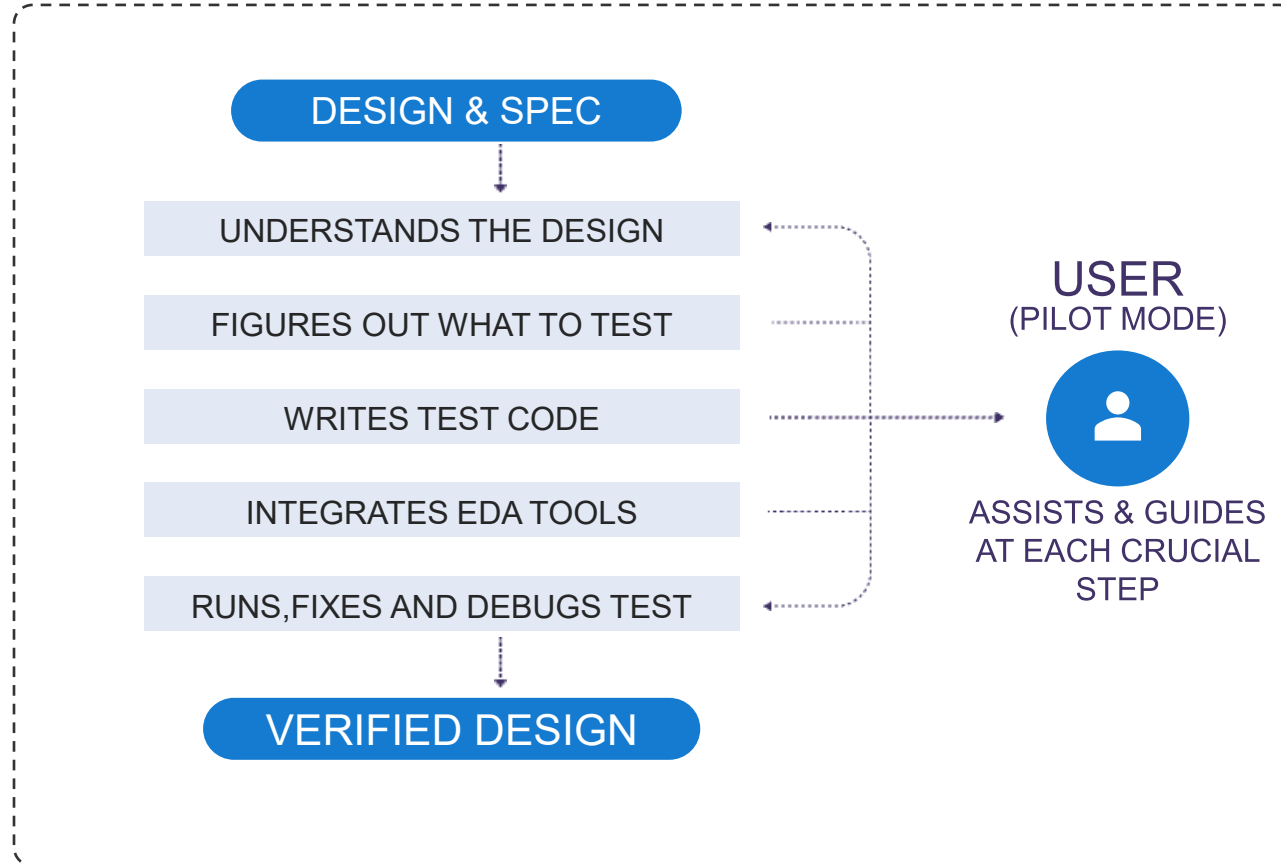
Waveform and coverage

From Design Understanding to Signoff — Fully Automated, Deeply Integrated

Reduce verification time by 10x • Scale your team instantly • Deploy in weeks, not months

Meet The World's First AI Chip Verification Engineer

Go from Design to 80% Coverage in Minutes!



Accelerating Verification Schedules By Over 70%

Deployed at leading customer sites, used on complex, industry-standard project

Understanding large designs & complex specifications



Mental Model Agent

Enable engineers to deeply understand the design intent

CHIP DESIGNERS

Deploying formal verification at scale for design/DV



Formal Verification Agent

Augment formal verification engineers to hunt harder-to-reach bugs

DESIGNERS FV ENGINEERS

Designer-driven unit testing to reduce early bugs



Unit Testing Agent

Enable RTL design teams to deliver high-quality RTL via ease of unit testing

DESIGNERS

Accelerating DV workflows and coverage closure



UVM Agent

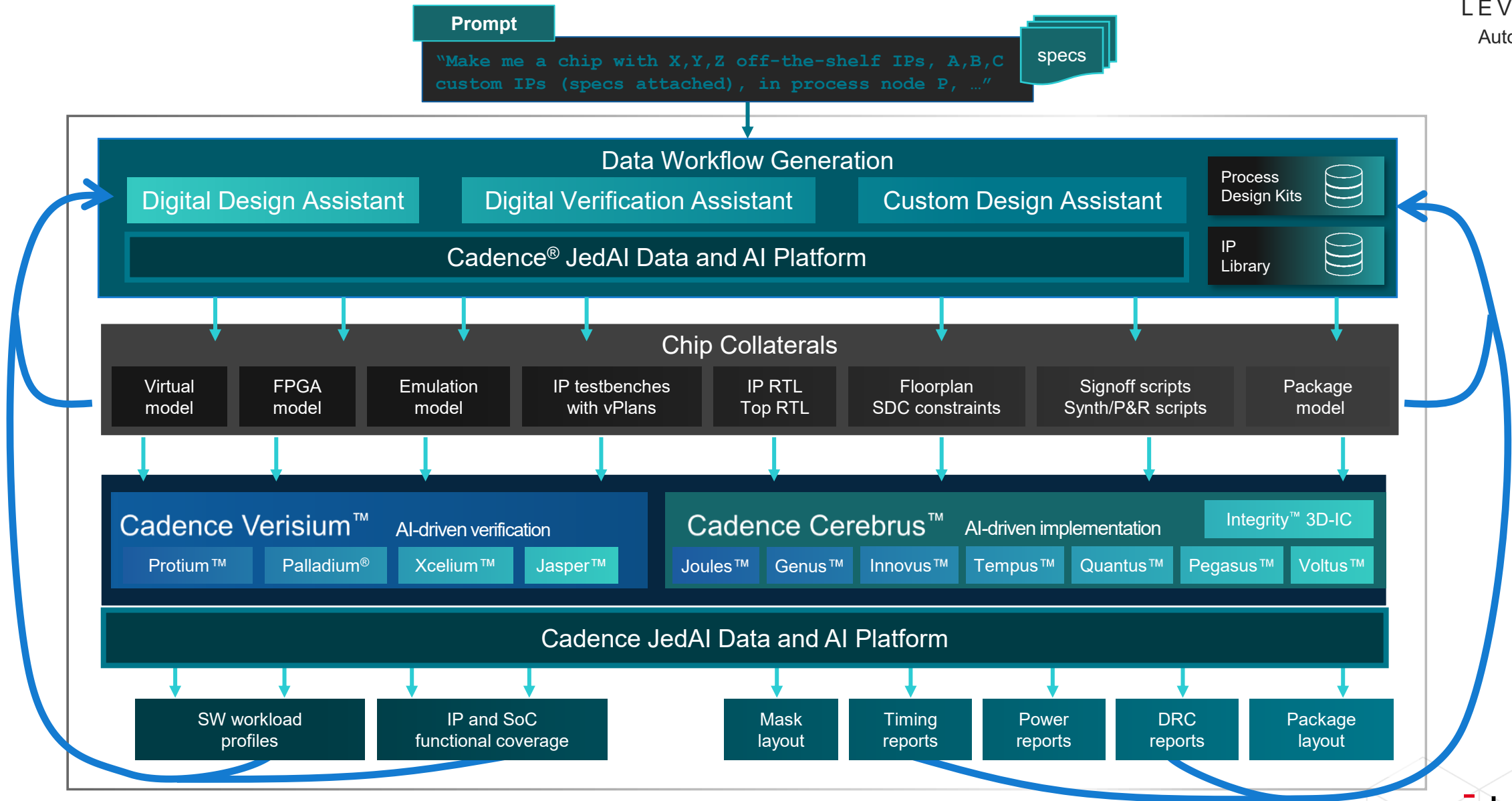
Testplans, functional covers, debugging, coverage closure, and more.

DV ENGINEERS AND DESIGNERS

Cadence Total Agentic AI for Silicon Design



LEVEL 5
Autonomy





cādence®

© 2025 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.

