2026
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SANTA CLARA, CA, USA
MARCH 2 - 5, 2026

SIGMA: Sign-off Intelligence with GenAI for Methodical Assurance in Formal Verification

R C Sanjay Krushnan
Moola Jeevan Chaitanya Goud
Sakthivel Ramaiah
Erik Seligman

cadence®

Agenda

- Introduction
- Problem Statement
- Solution/Innovation
- Implementation
- Case Study
- Results and Benefits
- Future Directions
- Summary

Introduction

- **Formal Verification (FV):** a critical methodology to ensure functional correctness
- **FV's key components include:**
  - Feature extraction & verification planning
  - Environment creation
  - Property development
- **Traditionally, these steps are manually intensive**
  - Require significant time, effort, and expertise
- **AI: Potential Solution**
  - In Cadence Jasper FV tools, **JedAI** platform + **Jasper® AI Assistant** are now available
  - We pioneered early versions of these tools

Problem statement

- **Extraction of the verification plan (vPlan).**
- **Formal environment creation.**
- **Developing System Verilog Assertions (SVAs).**

Dangers of manual steps in FV:

- **Time consuming**
- **Undetected bugs**

➔ **Compromised design integrity.**

➔ **Loss of customer trust**.

Need a reliable, automated solution that:

- **Ensures** comprehensive coverage
- **Automates** Formal Environment creation.
- **Accelerates** SVA generation for basic checks
- **Minimizes** risks of human error

Solution/Innovation: SIGMA

GenAI-Augmented Formal Verification Sign-Off flow: integrates JedAI + Jasper AI Assistant + custom scripts

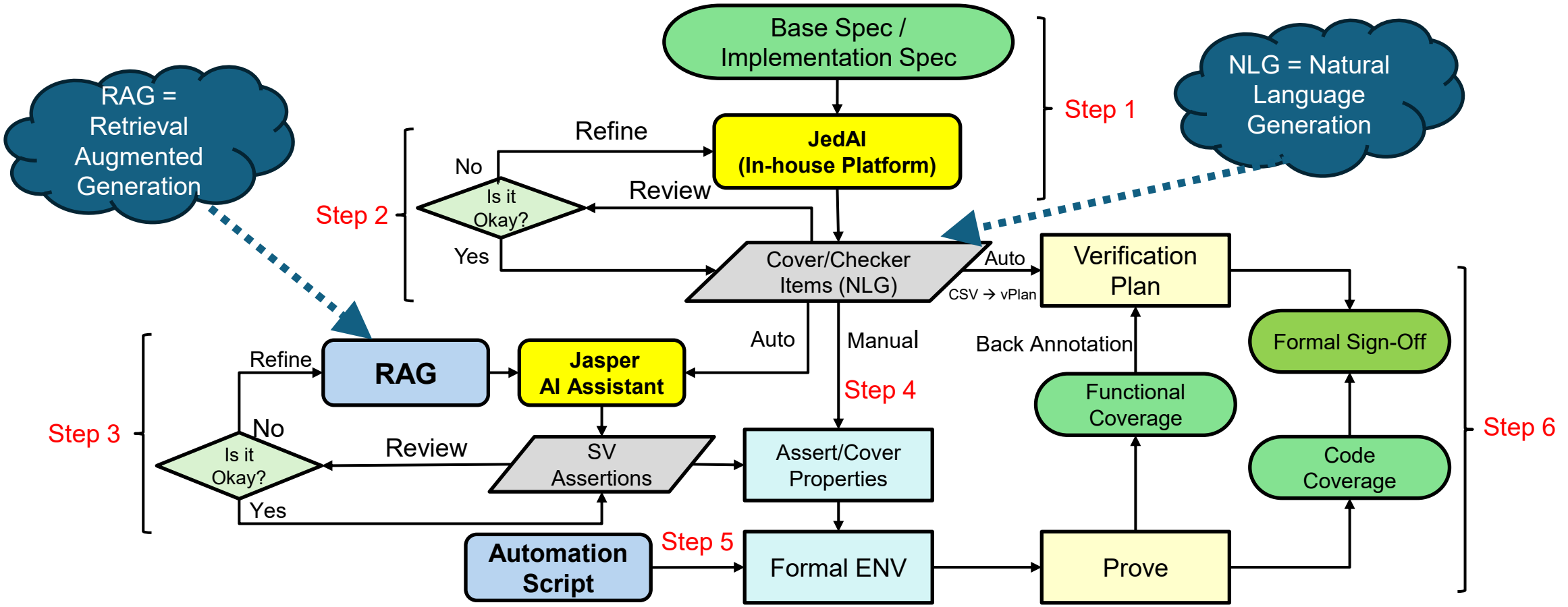
JedAI – A GenAI-based product development platform:
Automates the extraction of the verification plan (vPlan) from specifications.

Python – Automation scripts for creation of Formal Environment.

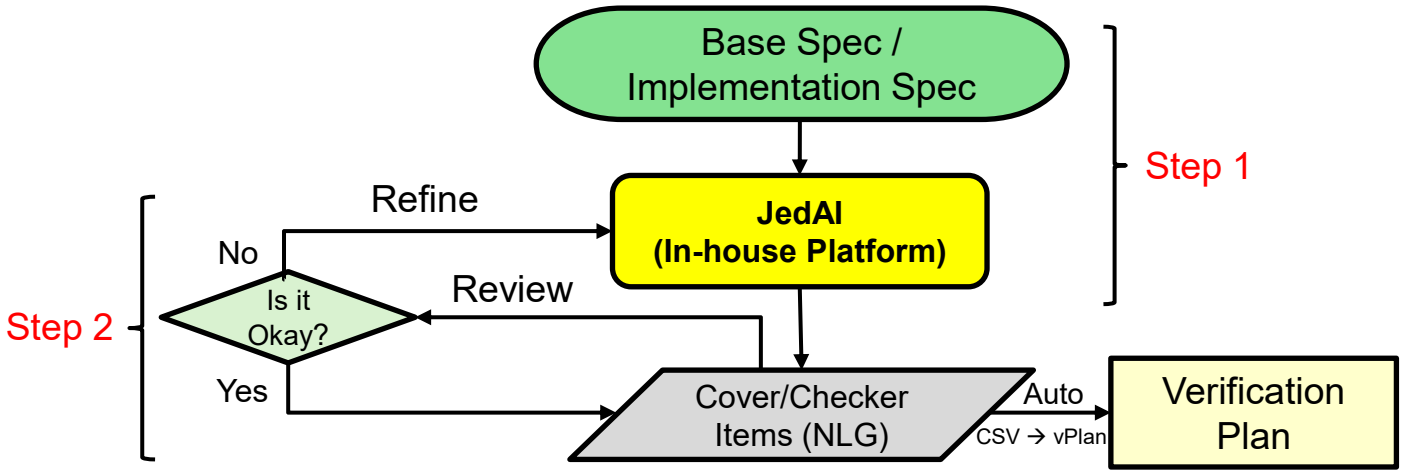
Jasper AI Assistant – A feature within the **Jasper Formal Verification Platform:**
Automatically generates System Verilog Assertions (SVAs).

This innovation streamlines the formal verification process, reduces the risk of bug escapes, and improves overall productivity in IP development.

GenAI-Augmented FV Sign-Off Flow



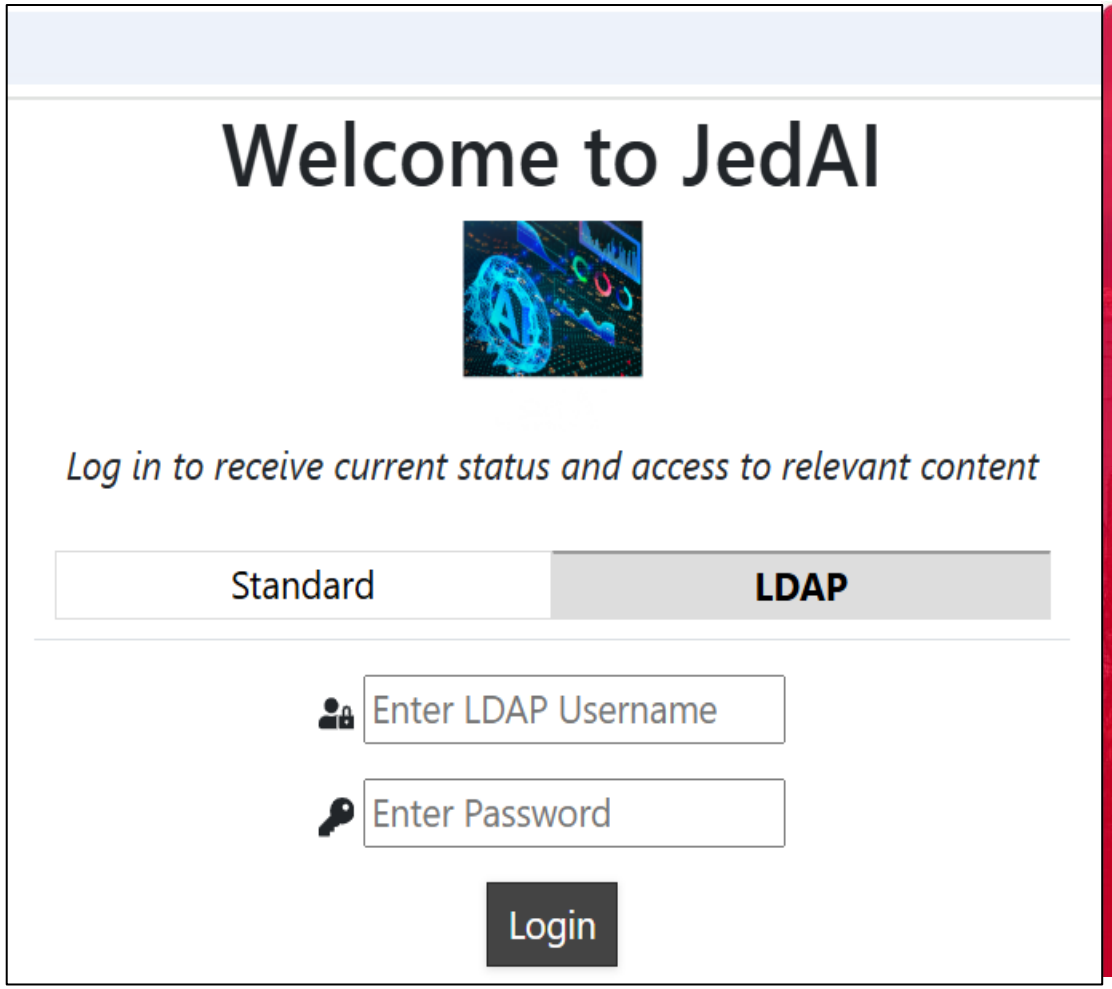
Implementation: Plan Generation

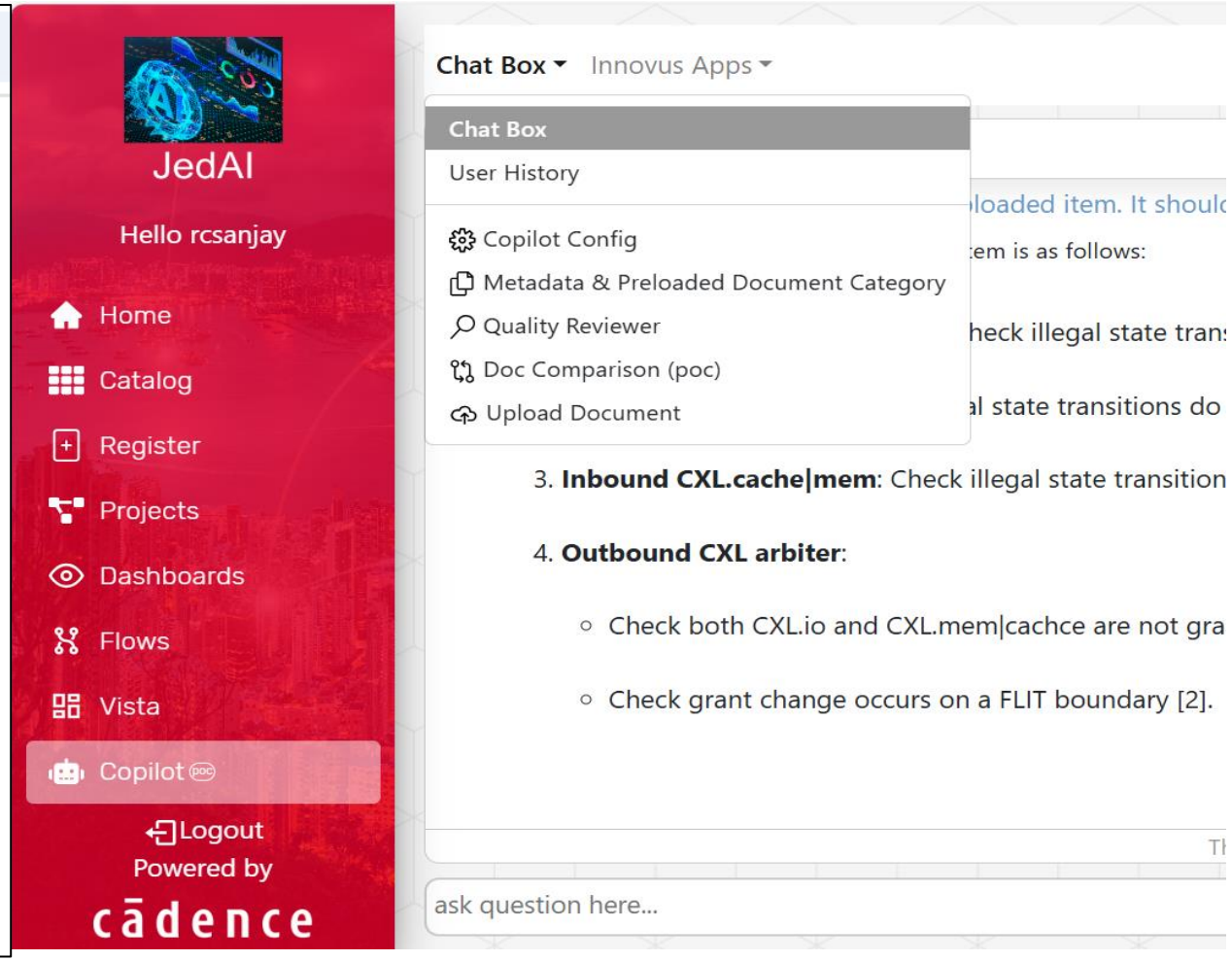


Specification Ingestion and vPlan Generation:
Input: Base/implementation specifications and reference Verification Plan
Process:

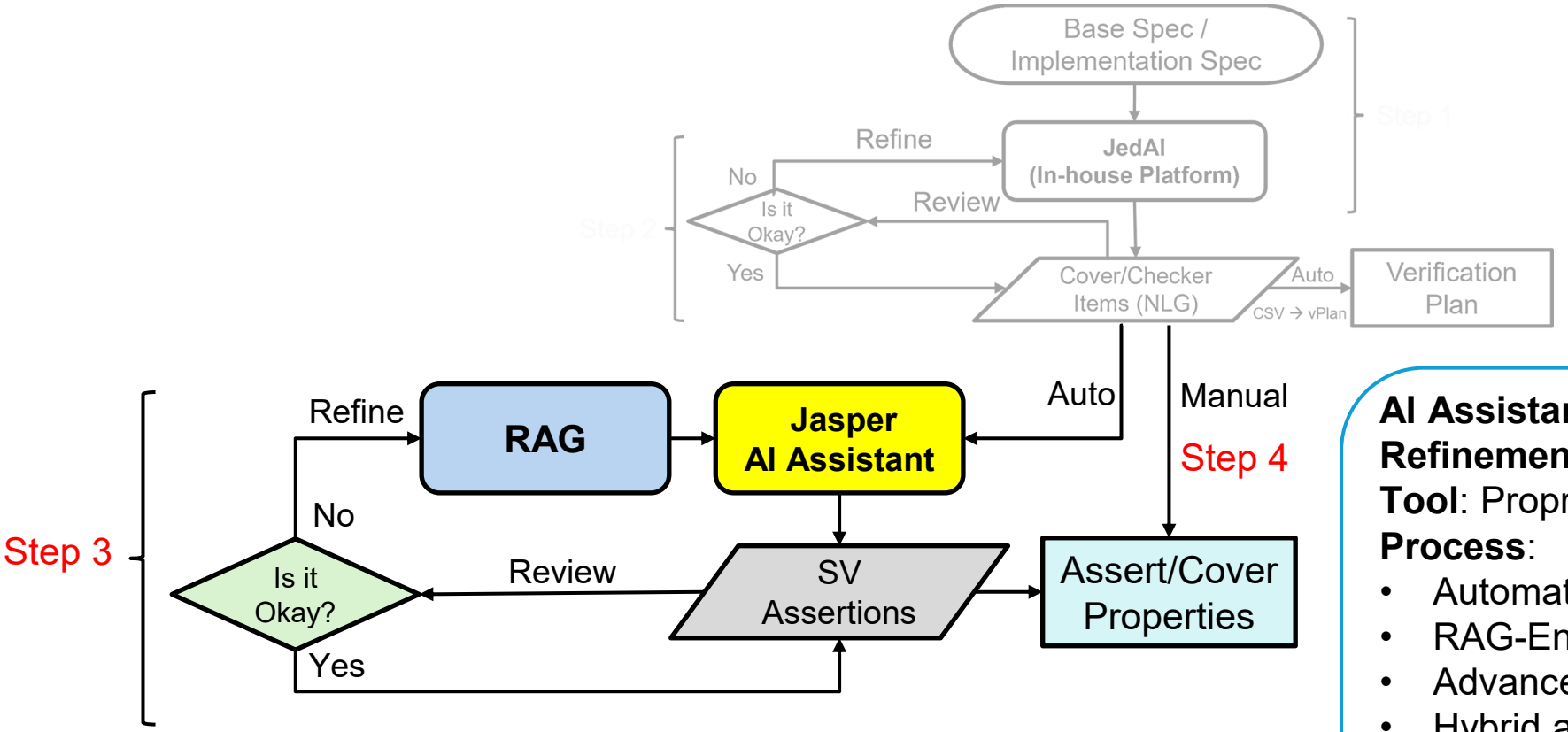
- GenAI uses NLP and LLM-based understanding to extract potential verification checks.
- Checks are manually reviewed for quality and completeness.
- GenAI converts the verification checks into a standardized verification plan format
- With help from reference examples

JedAI Platform





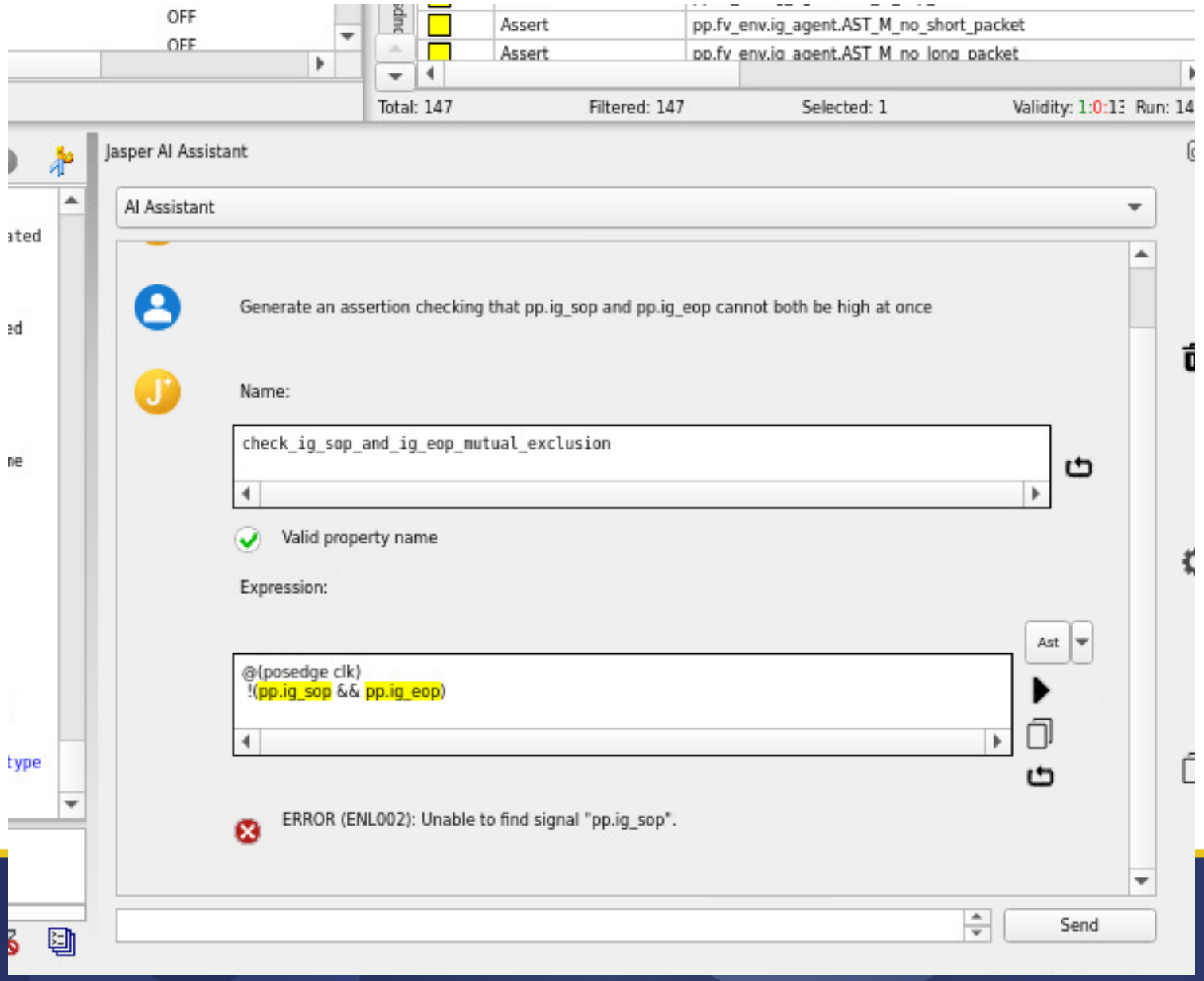
Implementation: SVA Generation



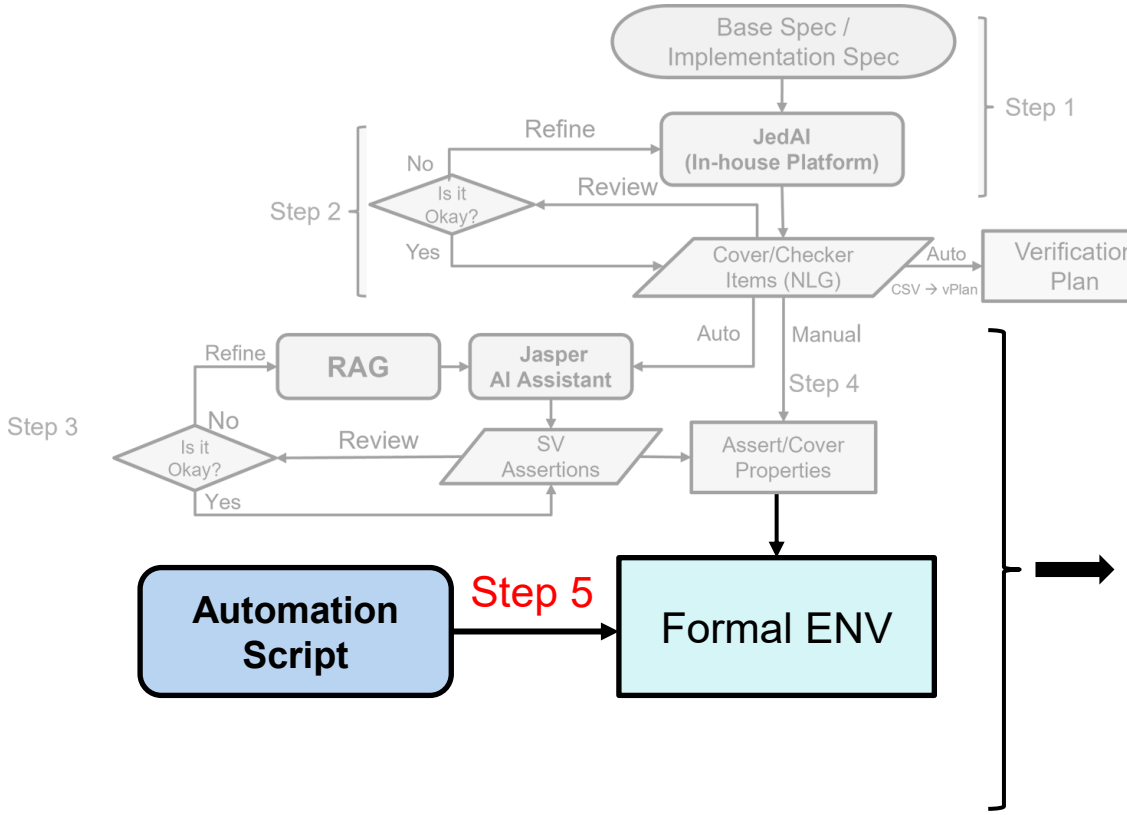
AI Assistant: SVA Generation and Refinement:
Tool: Proprietary AI Assistant
Process:

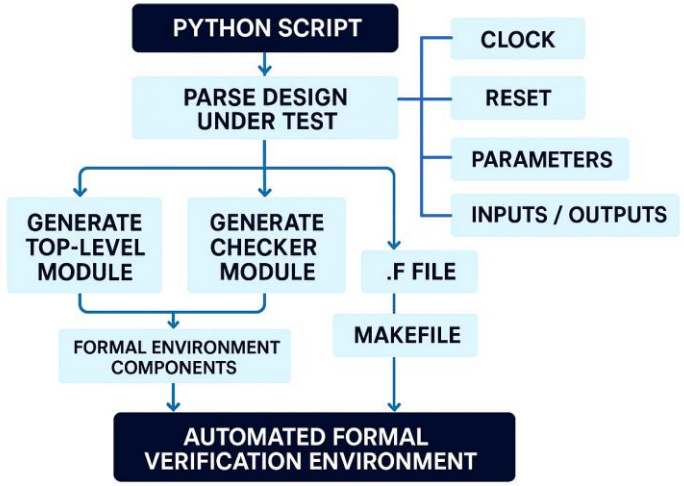
- Automatic SVA Generation - Basic Checks
- RAG-Enhanced Assistance - Medium Checks
- Advanced Checks coded manually
- Hybrid approach: Automation + Expert User

Jasper AI Assistant UI



Implementation: Environment Generation

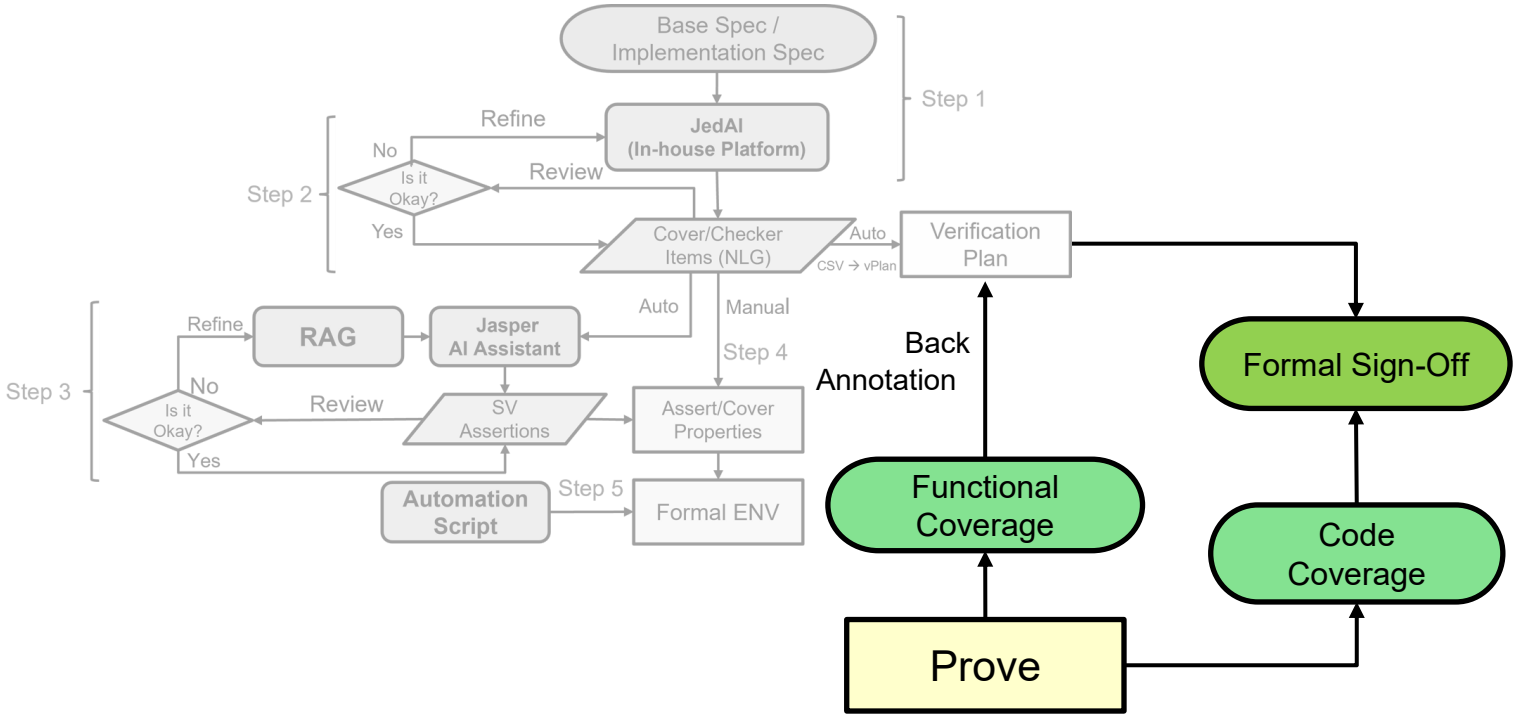




Scalable Formal Environment Generation:

- Environment generated with Python script
- Extracts DUT clocks, resets, parameters, I/Os.
- Generates the complete formal environment
  - Target design modules
  - Checker modules
  - File Lists
  - Makefile

Implementation: Plan Execution



Formal Proof Execution:

- Environment runs Jasper
- SVAs formally proven/falsified
- Failures reported for debug

Coverage Analysis:

- Functional Coverage analysis in Verisum Manager.
- Code coverage analysis using Jasper Coverage App + IMC.

CASE STUDY

PCIe 6.0® : Sequence Number Handshake

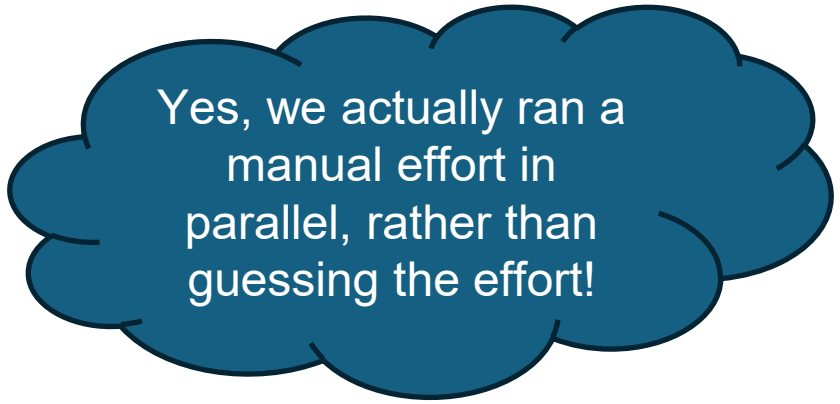
SIGMA Flow was evaluated on the PCIe Sequence Number Handshake block vs Traditional flow.

Design Overview:

- FSM manages flit sequence number alignment during PCIe link initialization.
- Four phases: IDLE, IDLE Flit Handshake, Sequence Number Handshake, and Normal Flit Exchange
  - With sequence tracking and Error detection.
- Ensures protocol compliance, robust error handling, and data integrity .

Experimentation:

- Exercised AI-based SIGMA flow for block-level FV
- Compared SIGMA flow with manual FV effort



CASE STUDY: Varying LLMs

LLM Model Deployment in GenAI Platform

- SIGMA verification flow incorporates three LLM models for the experimentation.
- LLMs enabled automated spec parsing and generation of verification artifacts, boosting
  - Efficiency,
  - Consistency
  - Scalability.

Example:

Query Prompt for Extracting Verification Checkers from Base/Implementation Specification

Formulate a formal verification strategy for the provided implementation specification, emphasizing the systematic extraction and definition of protocol-level checkers and functional coverage items. It must detail the construction of verification checkers targeting control signal sequencing, data integrity, and protocol compliance across all operational scenarios.

CASE STUDY: Comparing LLMs

S.No	LLM Model	Completeness Score	Quality Score	Advantage	Examples
1	LLM_1	80%	90%	High accuracy; excels in hierarchical plan generation and complex check classification	IDLE State - Entry Condition: Prove that the SM enters or remains in initial state when Link is down, or the link enters recovery. No Transmission: Prove that while in the initial state, no flits are transmitted. Exit Condition: Prove that if the SM is in initial state and ltssm enters L0 or Configuration.Idle or Recovery.Idle, the next state is IDLE FLIT Handshake PHASE.
2	LLM_2	75%	80%	Moderate performance; suitable for general-purpose extraction tasks	Idle Transition - If LTSSM enters L0 (or Configuration.Idle/Recovery.Idle) while in initial state, the SM moves to IDLE FLIT Handshake PHASE. IDLE Flit Exchange In IDLE FLIT Handshake PHASE, the transmitter must continuously send IDLE flits; any non-IDLE received flit is ignored.
3	LLM_3	65%	72%	Efficient parsing of specifications; good semantic analysis for basic checks	State Machine Transitions Entry to IDLE: Verify on Link is down or LTSSM recovery entry. IDLE to IDLE_FLIT_HS_PHASE: Verify transition occurs when LTSSM enters L0/Configuration.Idle/Recovery.Idle.

CASE STUDY: SVA Generation & RAG

- AI Assistant converts spec-derived checks into SVAs
- Needed RAG-based refinement for improved effectiveness
- Example:
  - Spec: From IDLE, if link_state_rec_idle or link_state_cfg_idle, transition to IDLE_FLIT_HS_PHASE.
  - Needed non-overlapping (|=>) for next-cycle transition.
  - AI used |-> incorrectly (same-cycle).
  - Manually Correct |=> stored via RAG → AI now generates proper SVAs.
  - Result: Spec-aligned and verified.

SVA COMPARISON

Manual SVA	(state == IDLE && (link_state_rec_idle || link_state_cfg_idle)) |=> (state == IDLE_FLIT_HS_PHASE)
AI Assistant SVA	(state == IDLE && (link_state_rec_idle || link_state_cfg_idle)) |-> (state == IDLE_FLIT_HS_PHASE)
RAG Refined SVA	(state == IDLE && (link_state_rec_idle || link_state_cfg_idle)) |=> (state == IDLE_FLIT_HS_PHASE)

CASE STUDY: Observations

- Streamlined Verification: GenAI-powered SIGMA enhanced and accelerated key flow steps.
- Left-Shift Gain: Enabled earlier verification, cutting FV effort by 25%.
- Productivity Boost: Engineers focused on complex tasks; routine work was automated.
- Scalable & Reusable: Methodology reused across blocks; consistent quality and low setup effort.
- Quality Proven: Results confirmed SIGMA matches traditional verification quality

CASE STUDY (contd...)

Bug Example

- Injected a bug into the DUT to evaluate the effectiveness of the SIGMA flow.
- Flow successfully detected the injected bug and generated the corresponding counterexample.
- A payload flit was incorrectly received with flit-usage 00, which is reserved for control flits (NOP/IDLE).
- As per protocol, payload flits must use flit-usage 01 to indicate valid data.

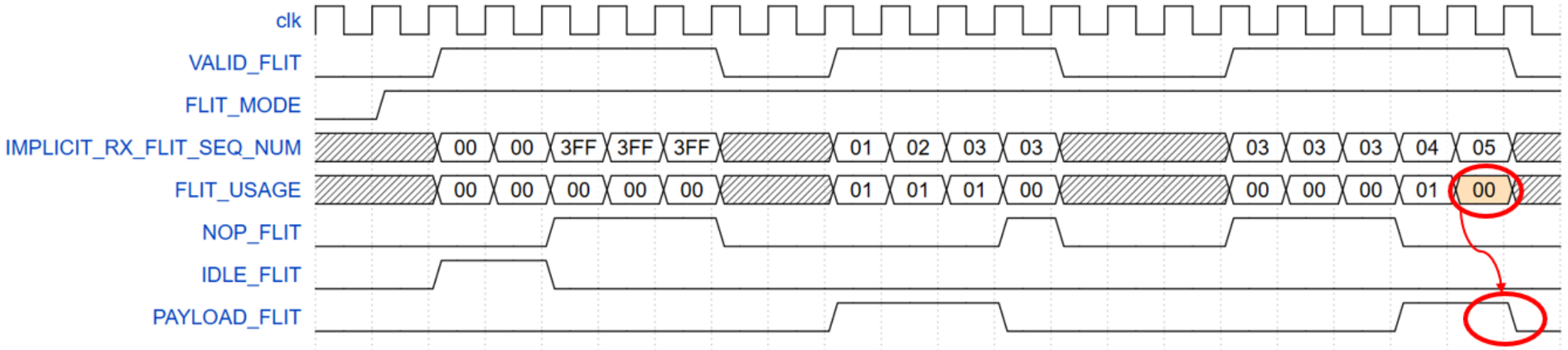


Illustration of Misinterpretation Flit Type

Results and Benefits

- The proposed flow significantly **accelerated PCIe Controller IP Verification** by enabling a **left shift** in the development cycle through enhancing:
  - **Accuracy** by minimizing missed checks
  - **Efficiency** by automating repetitive tasks
  - **Coverage** by ensuring all required checks are captured and verified

After success on SEQ_NUM, we used this flow for other blocks

S. No	Block	Gate Count	Effort(months)			
			Pre-Project Estimation	Traditional FV Signoff (Estimates except SEQ)	SIGMA Flow	SIGMA advantage
1	SEQ NUM FSM	Low	4	4 (Actual)	3	**25%**
2	ACK/NAK SCHEDULER	High	6	6	5	**17%**
3	TX RETRY BUFFER	Medium	6	6	4.5	**25%**
4	DLLP DECODER	Low	4	3	2	**33%**
5	DLP INSERT	Medium	5	5	3.5	**30%**

- Helped **fresh formal verification engineers** quickly learn:
  - How to extract features from specs
  - How SVAs are structured and applied
  - Enabled them to contribute to the project **much earlier** than traditional ramp-up timelines.
- Key benefits include faster time to market and reduced costs.

Future Directions

- Continue to leverage increasingly mature AI tools
  - Jasper AI Assistant, ChipStack Super AI Agent, …
- New capabilities will include:
  - Full flow automation from spec to testbench to signoff
  - More powerful SVA generation – including complex assertions
  - AI to address complexity & convergence
  - Intelligent debugging guided by AI agent

Summary

- **SIGMA AI-Augmented FV SignOff flow** combines automation with human oversight.
- **Verification Plan Generation from Spec** using an LLM + JedAI platform
- **SVA Generation**
  - **Basic checks:** Auto-converted to SVAs using Jasper AI Assistant.
  - **Medium checks:** Refined by RAG (Retrieval-Augmented Generation)
  - **Advanced checks**: Manually coded currently, Expect automation after tool improvements
- **Functional coverage and code coverage** are analyzed using vManager, Jasper, and IMC.
  - To ensure complete sign-off.
- This solution improves **efficiency, accuracy, and scalability** in the formal verification process.
- **SIGMA flow already pulls in timeline** by up to 33%, expected to grow with improved tools.