Revolutionizing Verification With GenAI-Powered Automation

A Paradigm Shift Towards Agentic Workflows

Dr Andy Penrose

April 2025

cadence®

The Agentic AI Opportunity

EDA 40-year History

Improving Productivity Through Design Abstraction

A new generation of Chip Design

Agentic AI

Excellence in Simulation and Optimization

NUMERICAL SOLVERS	SMT SOLVERS	GRAPH THEORY AND TOPOLOGY
BINARY DECISION DIAGRAMS	BOUNDED MODEL CHECKER	DISTRIBUTED PROCESSING
SAT SOLVERS	ADAPTIVE MESHING	NON-LINEAR SOLVERS
SYMBOLIC MODEL CHECKERS	MATRIX SOLVERS	COMPUTATIONAL GEOMETRY

Optimization AI

RTL Design Agent - Arbiter Design

Arbiter: A design clocked with *clk* and reset *rst*, has four requesting agents, each able to request a shared resource using one bit of *req* signal. The *gnt* signal indicates which agent is currently authorized to use the resource. There is also an *opcode* input, allowing certain commands that override the normal behavior of the arbiter, such as forcing a particular agent to get priority or cutting off access to the resource, for a period of time. There is an *op_error* output as well to signify that a bad opcode or illegal sequence of opcodes has been sent.

% *given this specification, generate a plan how to implement it in System Verilog*

Below is a plan with a detailed logical sequence for a SystemVerilog implementation:

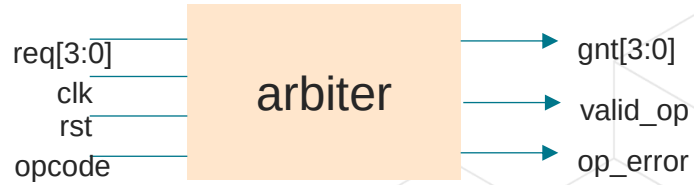
1. Define the top-level module and interface signals
2. Create an internal state machine
3. Normal arbitration logic
4. Opcode processing
5. Timing and separation of control
6. Error mechanism
7. Integration and verification
Feel free to ask for further details or help with coding specific parts of the module!

% *Generate SystemVerilog module interface definition using parameters*

Below is an example SystemVerilog interface definition for the given arbiter design:

```
module arbiter#(
  parameter int WIDTH = 4,                      // Number of request/grant lines
  parameter int HIGH_PRIORITY_INDEX = 0         // The index of the highest priority request (0 is highest)
)(
  input  logic            clk,
  input  logic            rst,                  // active high reset
  // Arbiter interface
  input  logic [WIDTH-1:0]  req,                // request signals vector
  output logic [WIDTH-1:0]  gnt,                // grant signals vector (one-hot encoded)
  // Opcode processing interface
  input  logic [7:0]        opcode,             // 8-bit opcode input
  output logic            valid_op,                                 // asserted when an allowed opcode is received
  output logic            op_error              // asserted when an illegal opcode is detected
);
```

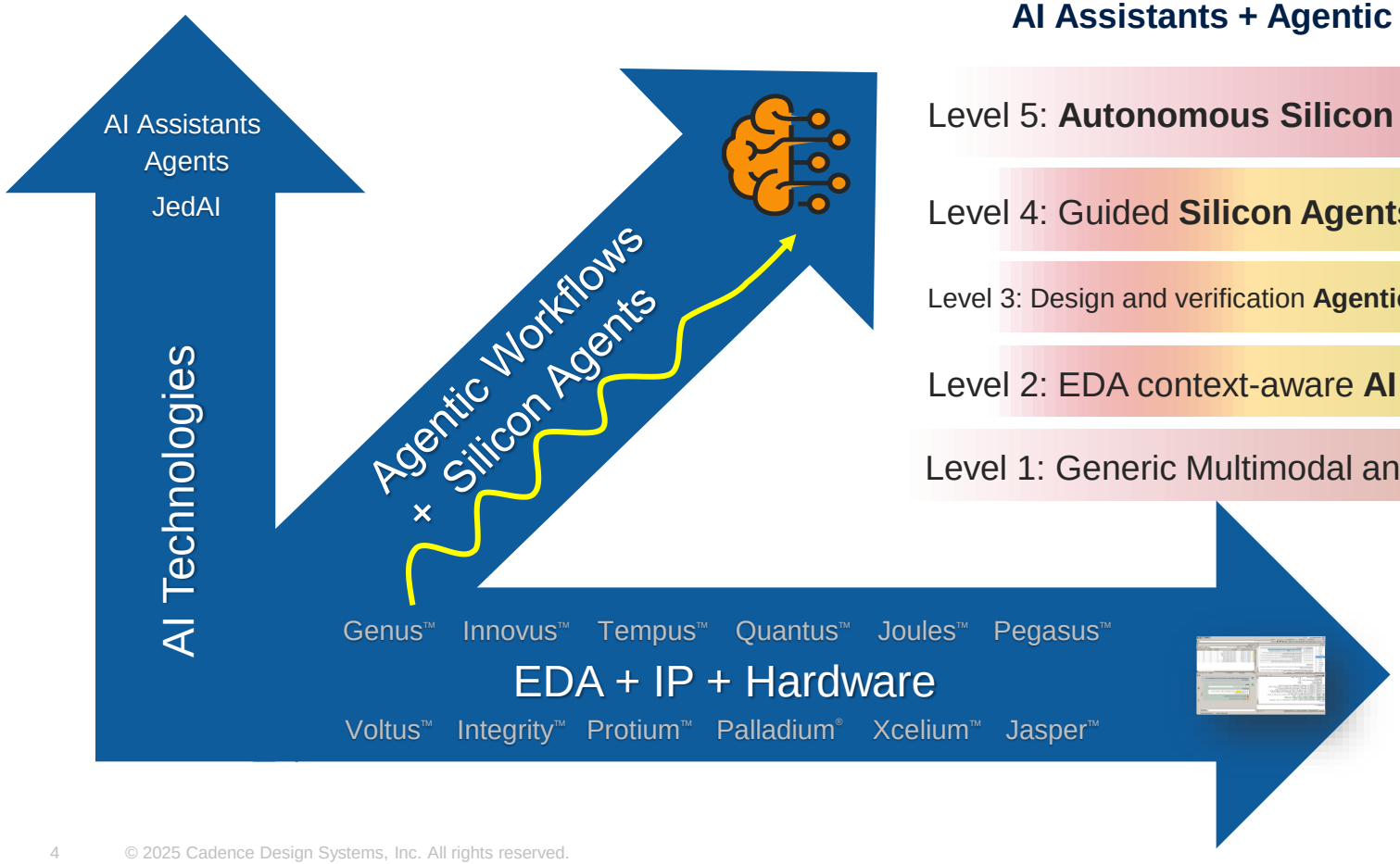
% *Draw top level schematics*



cadence

New EDA Vision: Silicon Agents for Chip Design

Leverage Artificial Intelligence technologies to seamlessly integrate with Cadence's AI-optimized software, hardware, and IP for creating data-driven Agentic Workflows and Silicon Agents



AI Assistants + Agentic Workflows + Silicon Agents

Level 5: **Autonomous Silicon Agents** for self innovation and design management

Level 4: Guided **Silicon Agents** (Specs, RTL, Testbenches, Netlist, …)

Level 3: Design and verification **Agentic workflows** (Verisium™, Cadence Cerebrus™, Optimality™ Explorer …)

Level 2: EDA context-aware **AI assistants** (Code and collateral generation, …)

Level 1: Generic Multimodal and Reasoning **LLM-**based Applications (search, doc, …)

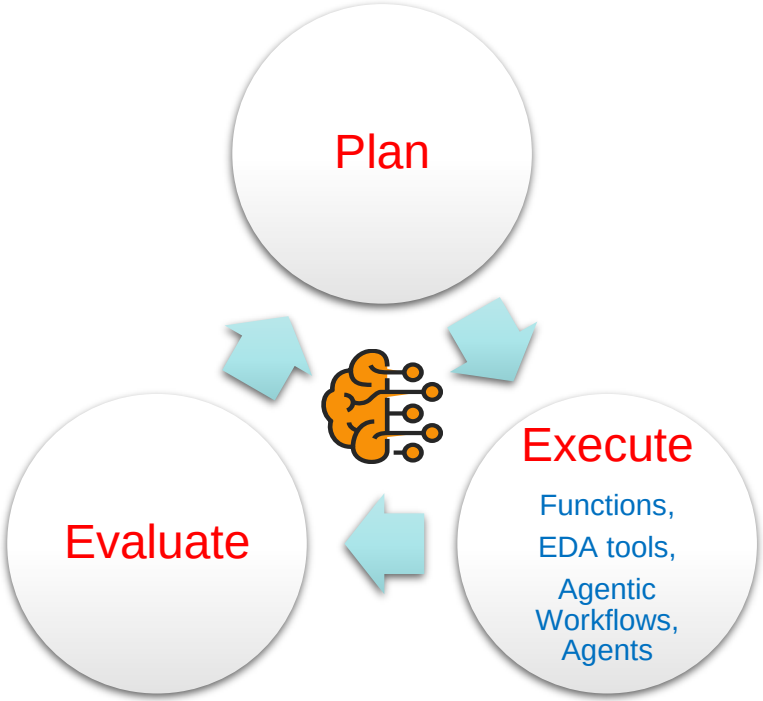
What Does “Agentic AI” Mean?

Autonomious and Competenece-Aware AI Software

Plan: understand its goals within a context and create or adapt an execution plan of single or multistep tasks to ensure successful goal attainment in dynamic conditions

Execute: dynamically and independently takes decisions to allocate resources, execute tasks, and possibly pass control to other agents

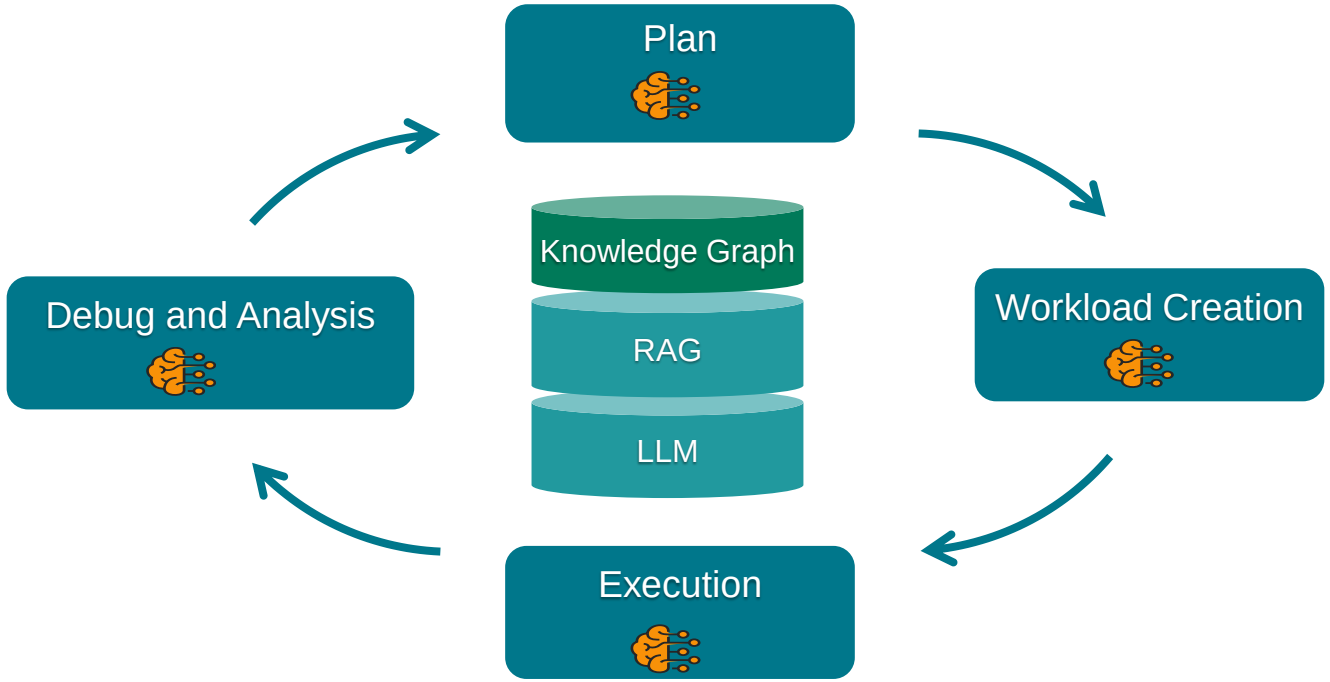
Evaluate: incrementally evaluates results against plan and context and streams out results



Cadence® JedAI Data and AI platform + AI Assistants

What Does “Agentic Workflow” Mean?

- Agentic AI technologies in traditional workflow tasks
- Integrated with AI Assistants
- Deterministic flow designed by experts, retaining the same methodology
- Aiming for productivity boost and better execution goals (PPA)
- Combining LLMs, AI Assistants, and guided AI Agents in selected tasks



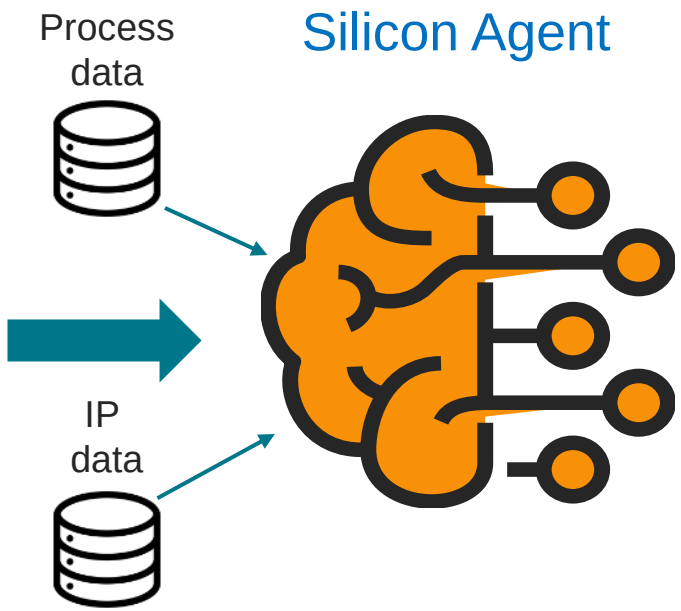
Cadence® JedAI Data and AI platform + AI Assistants

What Does “Silicon Agent” Mean ?

AI Agent for Chip Design

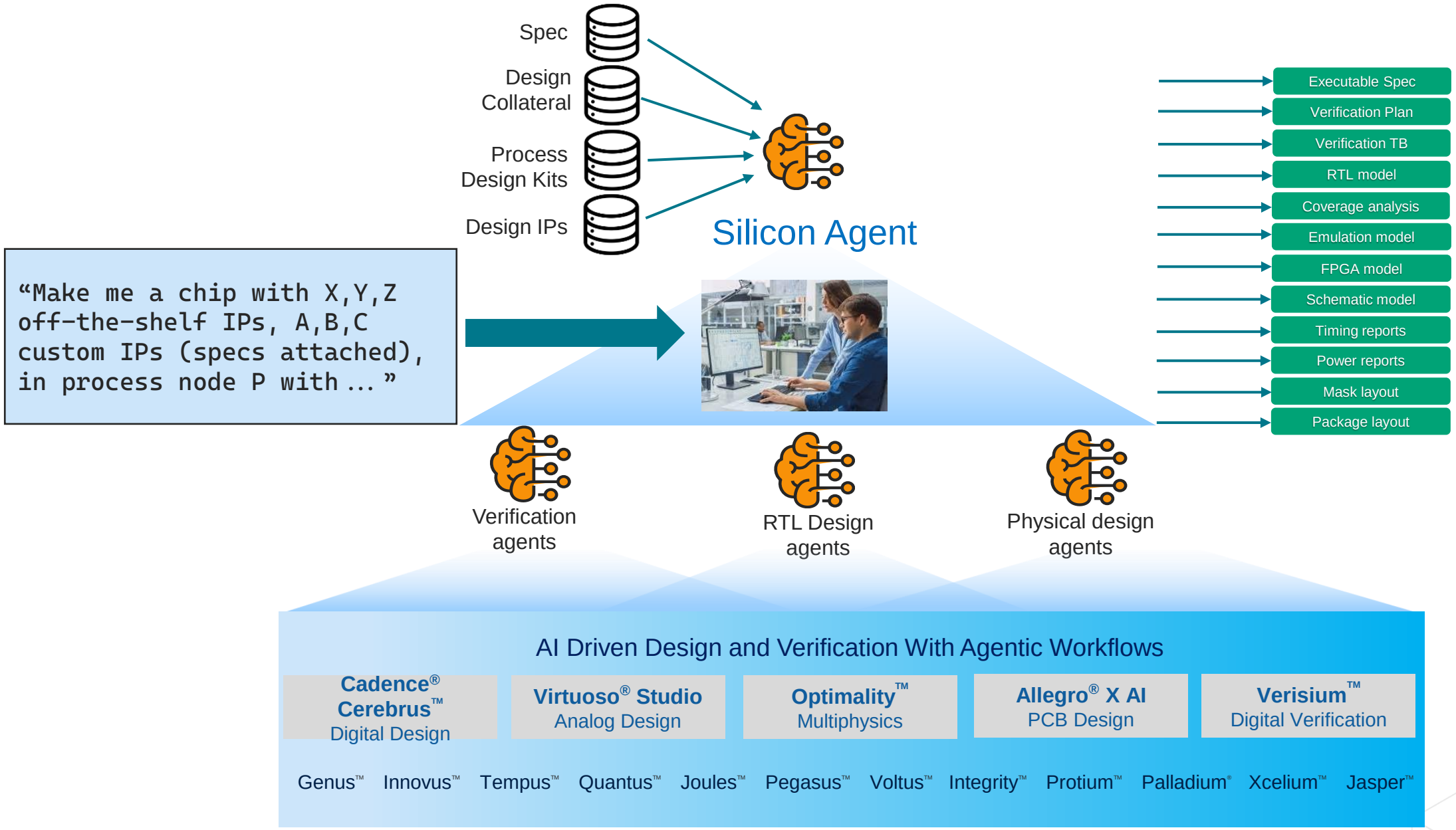
Prompt

“Make me a chip with X,Y,Z off-the-shelf IPs, A,B,C custom IPs (specs attached), in process node P with … ”



- Executable Spec
- Verification Plan
- Verification TB
- RTL model
- Coverage analysis
- Emulation model
- FPGA model
- Schematic model
- Timing reports
- Power reports
- Mask layout
- Package layout

Silicon Agent – Assisted by a Hierarchy of Domain-Specific Agents



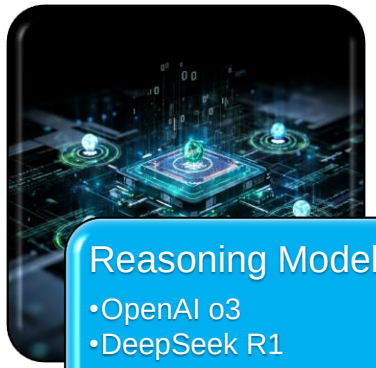
Capabilities Pipeline and Agentic Platform

Agentic Workflows and Silicon Agents

Cadence® JedAI Agentic AI Framework

Multimodal Foundation Models
- GPT
- Gemini
- Llama3

User-Aware Data and Collateral
- RAG
- Search
- LLM Tuning



Reasoning Models
- OpenAI o3
- DeepSeek R1
- Gemini 2.0 Pro
- Nemotron

Silicon Agents
- Design Tools
- Verification Tools
- Software Tools
- Data analytics tools
- Search Tools
- Planning and exec Tools

Learning from Data

Reinforcment Learning and Agents

Verification Agents – Functional Verification

Spec Analysis
- SpecMiner vPlan
- Verisium™ Manager

Planning Agents

Testbench generation
Assertion generation
Formal setup
Test management, selection, …
- SpecMiner
- VIP Assistant
- AutoFocus™

Workload Creation Agents

Smart job scheduling
Coverage driven optimization
Regression management
- SimAI™
- SmartProof
- SmartRun

Execution Agents

Bug localization
Root cause analysis
Failure prediction
- PinDown™
- AutoTriage™
- CodeMiner™
- WaveMiner™

Debug and Analysis Agents

Knowledge Graph
RAG
LLM

Protium™ Prototyping

Palladium® Emulation

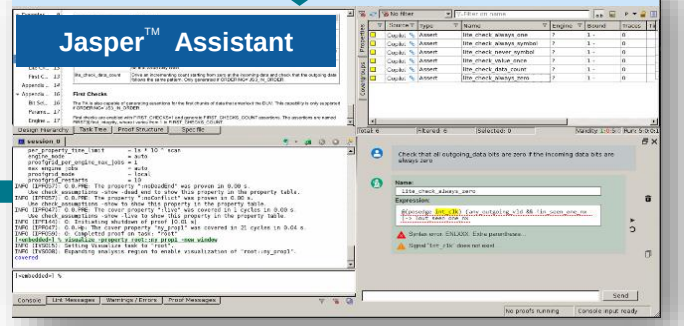
Xcelium™ Logic Simulation

Jasper™ Formal Verification

SpecMiner– Spec Analysis, Verification Plan, Testbench Gen

SpecMiner (SVA)

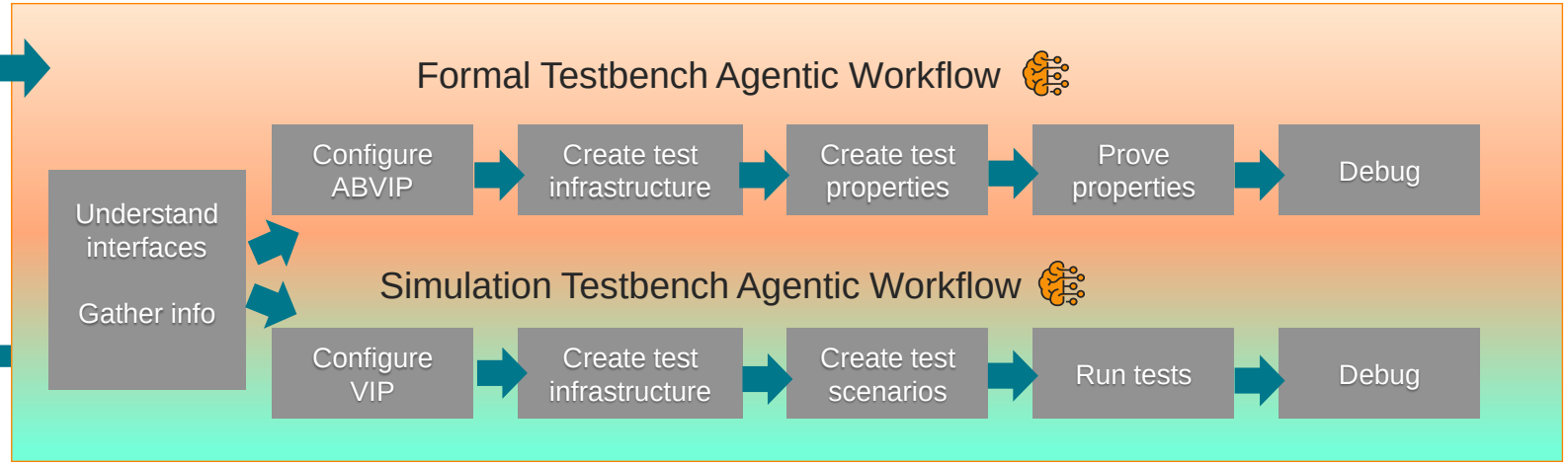
Generate SVA for: 'done' must be asserted in the next clock tick of 'clk' after signal 'sent' was asserted




```
assert property (@(posedge clk)
        $rose(sent) |=> done));
```

SVA Database

SpecMiner (Testbench)




```
class my_scoreboard extends uvm_scoreboard;
    `uvm_component_utils (my_scoreboard)
    function new (string name = "my_scoreboard", uvm_component parent= null);  …
```

Verification Plan Testbench Database