

PRAGYAN AI

Grow with Gyan

PragyanAI – SILICONAI

NITK | Stanford Ignite / Seed Spark | IIMA (Business Strategy Communication) | SLP Fellow |
25+ Years Product Exp | Tech Leadership | Serial Entrepreneur

Autonomous AI-First Engineering Workflows for Semiconductor Design, Verification & Talent Development
Democratizing Silicon Innovation for a Self-Reliant, AI-Powered India



Year of Incorporation / Registration
2021



Founder(s) & Location
Sateesh Ambesange & Uppin Rashmi
Bangalore, India



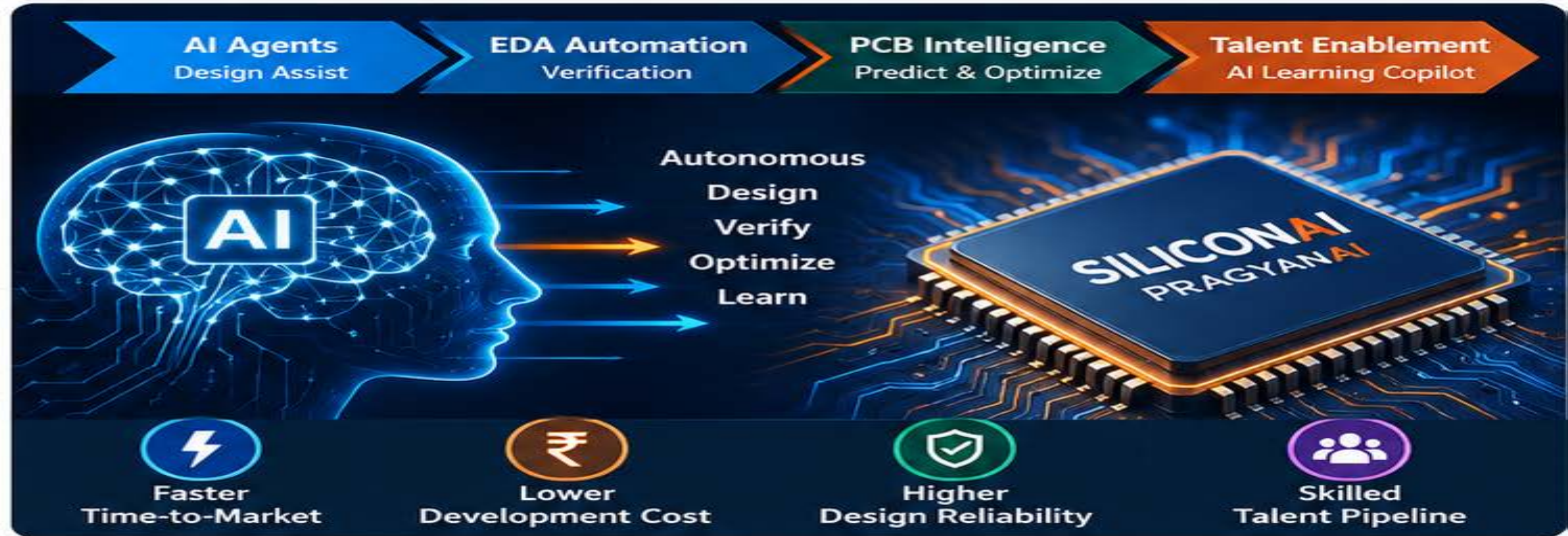
Website & Social Media Links
www.pragyansiliconai.com



Current Stage of Startup
TRL 4 : MVP
(Validated in Lab, Early Pilots)



Sector / Industry
**Semiconductor | EDA | PCB | AI |
DeepTech | Engineering Education**



12+ CORE SOLUTIONS

AI-POWERED TOOLS FOR THE FULL SILICON & ELECTRONICS LIFECYCLE



1. RTL Code Generator
AI-driven Verilog/SystemVerilog generation

Design Automation



2. Testbench Synthesis
Automated UVM testbench generation

Verification



3. AI Waveform Debugger
Root-cause analysis & waveform insights

Debug & Analysis



4. Formal Verification Assistant
Property generation & formal checks

Verification



5. PCB Design Copilot
AI-assisted schematic & layout design

PCB Design



6. PCB DFM/DFT Analyzer
Predictive manufacturability & test analysis

PCB Intelligence



7. Thermal & EMI Predictor
Pre-fabrication thermal and EMI analysis

Signal Integrity



8. Hardware-Aware AI Compiler
Model quantization for edge silicon

Edge AI



9. SoC Integration Assistant
IP integration & interface generation

SoC Design



10. Silicon Verification Insights
Coverage, risk & quality analytics

Analytics



11. Interactive Learning Copilot
AI tutor for VLSI/PCB/AI skills

Talent Development



12. Engineering Workflow Agent
End-to-end autonomous EDA workflows

Productivity

SpecLLM + Multi-Agent Engineering Specification System

It is an Engineering Specification Intelligence system that understands requirements, challenges ambiguity, discovers gaps, applies organizational knowledge, recommends engineering solutions, and produces traceable, verification-ready specifications.

Natural Language / Documents / Existing Specs



Multi-Agent AI System



Requirement Agent

- Extract functional & non-functional requirements
- Identify actors, inputs, outputs



Specification Analyst

- Structure & classify requirements
- Create engineering specification format



Constraint Agent

- Detect performance, power, area, latency, interface, security, standards constraints



Critic Agent

- Find ambiguities, contradictions
- Challenge assumptions
- Ask clarification questions



Gap Analysis Agent

- Identify missing information
- Analyze impact
- Suggest questions & data needed



Refinement Agent

- Resolve ambiguities
- Rewrite & enhance specification
- Define measurable parameters



Solution Intelligence Agent

- Recommend architectures
- Suggest design patterns, IPs
- Provide trade-off analysis



SPEC LLM Requirement Engine

- Understand
- Reason
- Analyze
- Use Knowledge
- Collaborate with Agents



Requirement Extraction

Extract & structure key requirements



Constraint Detection

Identify and analyze design constraints



Context / Intent Analysis

Understand context, goals and intent



Ambiguity Detection

Detect unclear, vague or conflicting requirements



Gap Analysis

Identify missing information and analyze impact



Specification Refinement

Refine, enhance and structure specification



Engineering Solution Recommendation

Suggest architectures, design options, IPs and best practices



Verified Engineering Specification

Complete, Consistent, Traceable, and Implementation-Ready



Engineering Knowledge Fabric



Company IP & Previous Projects

Design specs, architectures, lessons learned



Design Patterns & Reference Architectures

Reusable solutions and templates



Standards & Guidelines

Industry standards, coding & verification guidelines



Component / IP Library

Processors, interfaces, memories, PHYs, etc.



Design Rules & Best Practices

Technology, methodology, compliance rules



Datasheets & Manuals

Vendor IPs, tools, technology information



Expert Knowledge

Internal experts, forums, documentation



Outputs & Deliverables



Structured Engineering Specification (Verilog / SystemVerilog / VHDL friendly)



Traceable Requirements with IDs



Constraints and Assumptions



Interface, Behavior and Performance Specifications



Design Recommendations with Trade-offs



Reference Architectures and Reusable Components



Verification Considerations



Implementation Guidelines



Complete, Reviewed & Verified Specification



1 Requirement Agent Input → Understand

Understands natural language requirements and converts them into structured, traceable functional and non-functional requirements.

INPUTS

- Natural Language** (Any format)
"Design a high-speed data acquisition module for sensor data"
- Documents** (PDF, DOC, PPT)
PRD, SRS, emails, meeting notes, etc.
- Existing Specs**
Previous projects, legacy specifications
- User Intent & Context**
Use case, domain, constraints, goals



Requirement Agent

OUTPUTS

- Functional Requirements (FR)**
What the system should do
- Non-Functional Requirements (NFR)**
Performance, power, area, reliability, etc.
- Actors, Inputs, Outputs, Behaviors**
- Structured Requirements**
Clear, unambiguous requirements
- Requirement IDs & Traceability**
Unique IDs and traceable links

KEY CAPABILITIES

- ✓ Parse natural-language requirements
- ✓ Extract functional requirements
- ✓ Extract non-functional requirements
- ✓ Identify actors, inputs, outputs and behaviors
- ✓ Convert unstructured text into structured requirements
- ✓ Maintain requirement IDs and traceability
- ✓ Handle multiple document formats
- ✓ Ask clarification questions (if needed)

2 Specification Analyst Understand → Structure

Converts extracted requirements into engineering terminology, classifies them, identifies dependencies and generates structured specifications.

INPUTS (FROM REQUIREMENT AGENT)

- Structured Requirements**
Functional & Non-functional requirements
- Actors, Inputs, Outputs**
System elements & behaviors
- Requirement IDs**
Traceability information
- Initial Context**
Domain, constraints, use case



Specification Analyst

OUTPUTS

- Engineering Specification**
Structured, detailed spec document
- Classified Requirements**
Functional, non-functional, interface, timing, etc.
- Requirement Dependencies**
Inter-relationships and hierarchies
- Missing Sections**
Identify gaps and missing information
- Requirement-to-Spec Mapping**
Traceability matrix

KEY CAPABILITIES

- ✓ Convert requirements into engineering terminology
- ✓ Classify requirements
- ✓ Identify dependencies
- ✓ Generate structured specification
- ✓ Detect missing sections
- ✓ Maintain requirement-to-specification mapping
- ✓ Ensure completeness and consistency
- ✓ Provide recommendations and next steps



3 Constraint Agent Find what limits the design

Analyzes the requirements to identify and extract all design constraints that will impact the architecture, implementation and verification.

INPUTS (FROM PREVIOUS AGENTS)

- Structured Requirements
- System Context and Use Cases
- Performance Goals
- Standards & Guidelines
- Previous Projects / Company IP



IDENTIFIED CONSTRAINTS

- Performance**
Speed, frequency, throughput
- Power**
Power budget, low power modes
- Area**
Silicon area, utilization limits
- Latency**
Response time, cycle latency
- Throughput**
Data processing rate
- Memory**
Memory size, bandwidth
- Interface / Protocol**
Protocol standards, IO, interfaces
- Technology / Process**
Process node, technology limits
- Safety / Security / Compliance**
Functional safety, security, industry standards (ISO, IEC, etc.)
- Timing Constraints**
Setup/hold, clocking, timing closure

EXAMPLE OUTPUT

- Max clock frequency: 500 MHz
- Latency: < 10 μ s
- Throughput: 10 Gbps
- Power budget: < 500 mW
- Area: < 2 mm²
- Memory: 1 MB on-chip
- Interface: AXI4, PCIe Gen4
- Technology: 7nm
- Security: AES-256, secure boot
- Compliance: ISO 26262
- Timing: Setup/Hold per STA

KEY CAPABILITIES

- Extract explicit and implicit constraints
- Identify quantitative limits
- Map to industry standards
- Check technology feasibility
- Highlight trade-offs
- Validate consistency with requirements
- Maintain constraint traceability
- Identify missing constraint information
- Provide recommendations
- Enable architecture and design decisions

4 Critic Agent Challenge the specification

Critically analyzes the requirements and specifications to find issues, ambiguities, contradictions and missing information, and asks the right questions.

INPUTS (FROM PREVIOUS AGENTS)

- Structured Requirements
- Identified Constraints
- Initial Specification
- Design Context
- Standards & Guidelines



ANALYSIS RESULTS

- Contradictions**
Conflicting requirements
- Ambiguities**
Unclear or vague statements
- Unrealistic Requirements**
Not feasible or impractical
- Missing Assumptions**
Unstated assumptions
- Design Challenges**
Challenge design decisions
- Risk Analysis**
What could go wrong?
- Clarification Questions**
Specific questions to the user
- Improvement Suggestions**
Recommendations for better specs

EXAMPLE OUTPUT (CRITIQUE)

- Contradiction:** Latency < 1 μ s conflicts with power budget of 100 mW.
- Ambiguity:** "Process data quickly" – what is the exact latency/throughput requirement?
- Missing Assumption:** Input data size not specified.
- Unrealistic:** 10 Gbps throughput may not be achievable in 7nm with given power budget.
- Clarification Question:** What is the maximum data size per transaction?

KEY CAPABILITIES

- Find contradictions and conflicts
- Detect ambiguous statements
- Identify unrealistic requirements
- Detect missing assumptions
- Challenge design decisions
- Ask "What could go wrong?"
- Generate clarification questions
- Suggest improvements
- Validate completeness and consistency
- Provide risk analysis and recommendations

Iterative Feedback Loop → Refine Requirements and Specifications



5



Gap Analysis Agent

Find what's missing

Analyzes requirements to identify missing information, compares with expected engineering details, assesses impact and suggests clarification questions.



EXAMPLE: "The system should process data quickly."

Aspect	Expected Engineering Information	Available Information	Missing Information	Impact	Clarification Questions / Recommendation
Latency	Maximum latency, average latency, worst-case latency	Not specified	Latency value, conditions	Affects architecture, buffering, pipeline depth	What is the maximum acceptable latency? Is this average or worst-case?
Throughput	Data rate/throughput (e.g., Mbps, Gbps, transactions/sec)	Not specified	Throughput value, peak vs sustained	Affects interface, memory, processing resources	What throughput is expected? Peak or sustained?
Input Data Size	Data size per transaction/frame	Not specified	Input data size, format, bit width	Affects memory, buffer size, interface design	What is the input data size and format?
Clock Frequency	Clock frequency (e.g., MHz, GHz)	Not specified	Target frequency, process technology	Affects timing, performance, power	What is the target clock frequency?
Power	Power budget (e.g., mW, W)	Not specified	Power constraints, power modes	Affects architecture, low-power design	What is the power constraint?
Worst-case Scenario	Worst-case conditions (e.g., max data, high temperature)	Not specified	Operating conditions, worst-case scenario	Affects reliability, design margins	What are the worst-case scenarios (data, temp, voltage, etc.)?

KEY CAPABILITIES

- ✓ Identify missing engineering information
- ✓ Compare with industry standards and templates
- ✓ Assess impact of missing information
- ✓ Generate targeted clarification questions
- ✓ Analyze functional and non-functional gaps
- ✓ Cover performance, power, area, timing, interfaces
- ✓ Prioritize gaps by impact
- ✓ Support iterative refinement with stakeholders

6



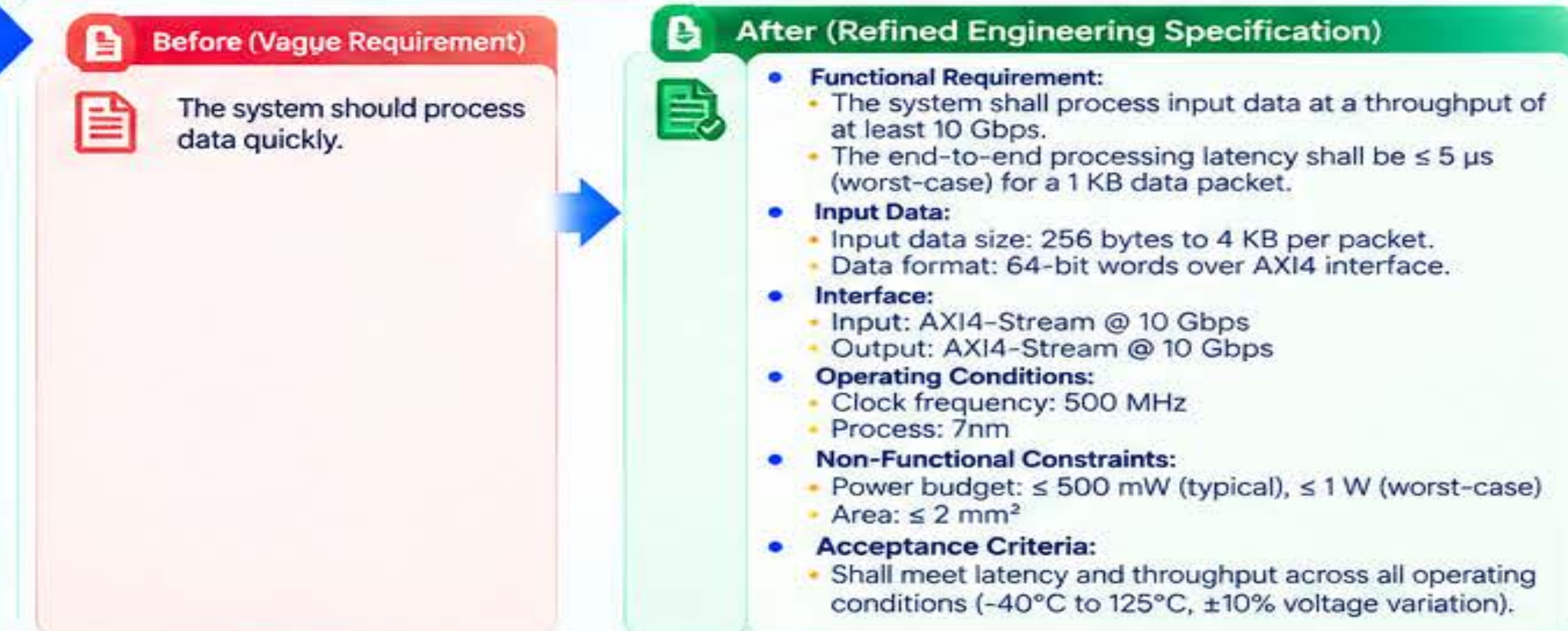
Refinement Agent

Turn incomplete requirements into engineering-grade specifications

Refines requirements by adding measurable parameters, resolving ambiguities, defining assumptions and generating structured, verifiable specifications.



EXAMPLE: Refinement of "The system should process data quickly."



KEY CAPABILITIES

- ✓ Rewrite vague requirements into precise, measurable statements
- ✓ Add quantitative parameters and ranges
- ✓ Resolve ambiguous terms and use standard terminology
- ✓ Define explicit assumptions
- ✓ Structure interfaces, inputs and outputs
- ✓ Define expected behavior and error handling
- ✓ Generate acceptance criteria for verification
- ✓ Produce complete, engineering-grade specifications

7 Solution / Engineering Intelligence Agent

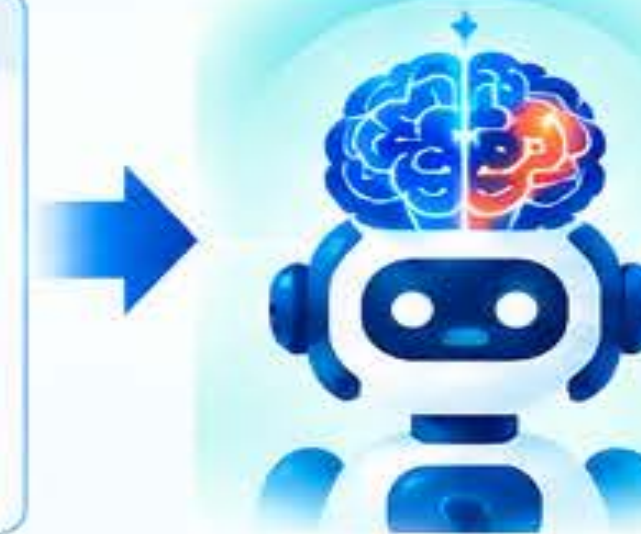
Given this requirement, how should an engineer solve it?



Example: Design a High-Speed Data Acquisition Module

Refined Requirement (from previous agents)

- Acquire analog sensor data at high speed
- Throughput: ≥ 10 Gbps
- Latency: $\leq 5 \mu s$ (worst-case)
- Input data: 12-bit, 1 GSPS
- Multi-channel: 4 channels
- Target platform: FPGA (option for ASIC)
- Power budget: ≤ 5 W
- Interface: PCIe Gen4 for data transfer
- Operating environment: $-40^{\circ}C$ to $125^{\circ}C$



Engineering Intelligence Agent

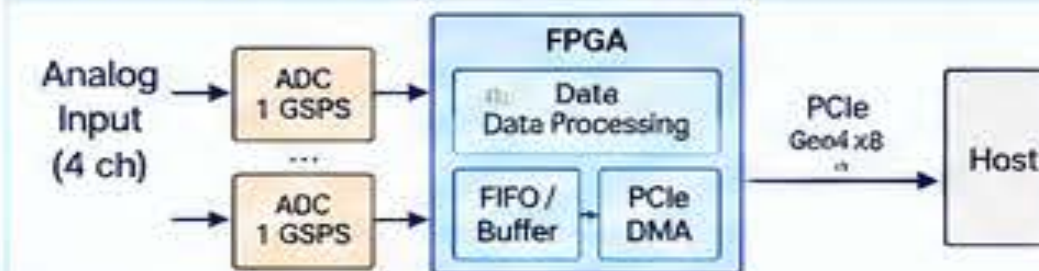
- Retrieves relevant knowledge
- Finds similar designs and IPs
- Applies design patterns & standards
- Generates multiple architectures
- Performs trade-off analysis
- Recommends the best solution

Retrieved Knowledge & References



Solution Options Generated

Option 1: FPGA-based Architecture



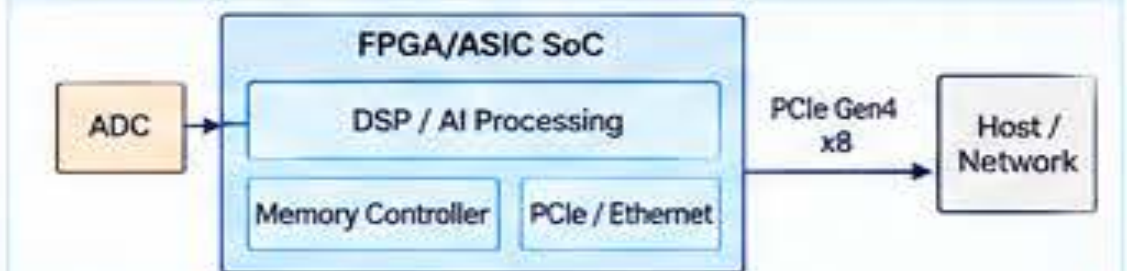
- High flexibility, faster development
- Uses commercial ADC + PCIe IP
- Suitable for prototyping and mid-volume

Option 2: ASIC-based Architecture



- Optimized performance, power and area
- Best for high-volume production
- Customizable for application needs

Option 3: SoC / Heterogeneous



- Integrated acquisition + processing
- Supports real-time analytics
- Higher complexity, higher cost

Trade-off Analysis

Criteria	FPGA Option	ASIC Option	SoC / Heterogeneous
Performance	High	Very High	High
Latency	Low (μs)	Very Low	Low
Power	Medium (3–5 W)	Low (1–3 W)	Medium
Area	Large	Small (optimized)	Medium
Development Time	Short	Long	Medium
Cost (NRE)	Low	High	Medium
Flexibility	High	Low	Medium
Best Use Case	Prototype / Low-Mid Volume	High Volume	Advanced Applications (AI/DSP)

Recommended Solution

Recommended Architecture



Key Components / IPs

- High-speed ADC (e.g., AD9254/AD9680)
- FPGA (e.g., Xilinx Ultrascale+)
- PCIe Gen4 IP (Xilinx/Intel)
- DDR4/DDR5 memory controller
- AXI-Stream, DMA, FIFO IP

Interface & Protocols

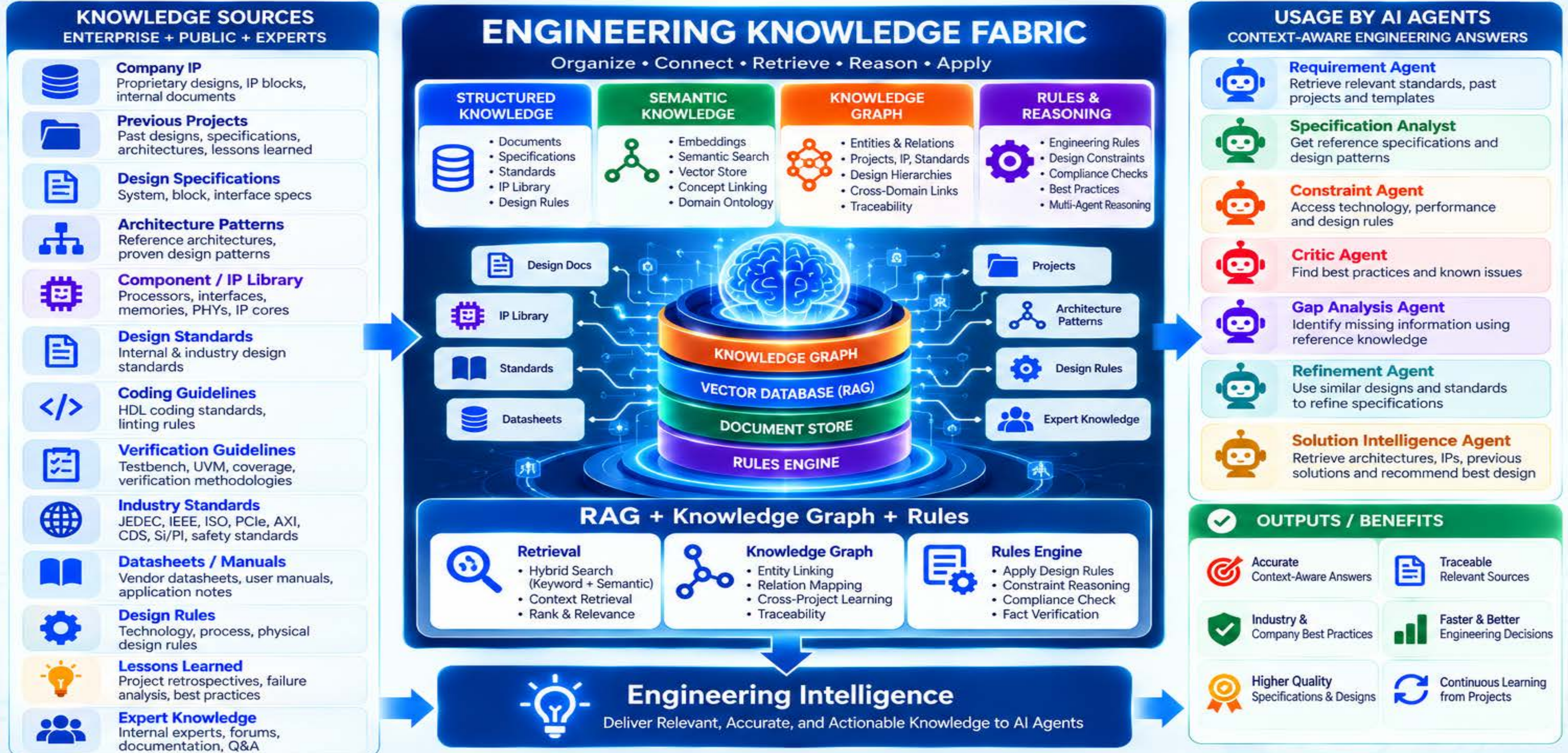
- ADC interface (JESD204B/C)
- AXI4-Stream / AXI4
- PCIe Gen4 x8
- Control interface (AXI-Lite)

- Meets performance (≥ 10 Gbps)
- Latency $\leq 5 \mu s$
- Uses proven ADC and PCIe IP
- Scalable to ASIC in future
- Supports multi-channel acquisition
- Complies with industry standards
- Low development risk

Next Steps

- Validate architecture with simulation
- Evaluate ADC and FPGA IP
- Develop RTL and testbench
- Prototype and performance testing
- Plan ASIC migration (if required)

Unified Engineering Knowledge for Smarter, Faster and Higher Quality Design Specifications



Your Company Knowledge. Your Standards. Your Solutions. Powered by AI Agents.



Bring Your Company Knowledge

Add your IP, projects, standards, design guidelines, documents



Customised & Fine-tuned

Adapt to your processes, terminology and design style



Your Unique Solutions

Learns from your past projects to provide personalised, high-quality solutions



Multi-Agent Intelligence

Retrieval + Reasoning + Analysis + Customised Recommendations



Continuous Learning

Gets smarter with your new projects, feedback and data

1 YOUR COMPANY KNOWLEDGE BASE

Add, organize and manage your engineering knowledge



Company IP

Reusable IP blocks, proprietary designs, internal documents



Previous Projects

Past designs, specifications, architectures, lessons learned



Design Specifications

System, block, interface specifications



Architecture Patterns

Reference architectures, design patterns



Component / IP Library

Processors, interfaces, memories, PHYs, IP cores



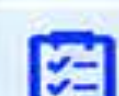
Design Standards

Internal coding, design and verification standards



Coding Guidelines

HDL coding standards, linting rules



Verification Guidelines

UVM, testbench, coverage, verification methodologies



Industry Standards

JEDEC, IEEE, ISO, PCIe, AXI, CDC, etc.



Datasheets / Manuals

Vendor datasheets, user manuals, application notes



Design Rules

Technology, process, physical design rules



Lessons Learned

Project retrospectives, failure analysis, best practices



Expert Knowledge

Internal experts, forums, documentation, Q&A

CUSTOM UPLOAD



PDF



DOC



PPT



TXT



CODE



CSV

Upload your documents, wikis, code, spreadsheets, etc.

2 ENGINEERING KNOWLEDGE FABRIC

Unified, connected and intelligent knowledge layer for your company

Document Store



- Documents
- Specifications
- Project reports
- Manuals / Datasheets
- Internal wikis

Semantic Indexing



- Embeddings
- Semantic search
- Metadata
- Domain ontology
- Vector database

Knowledge Graph

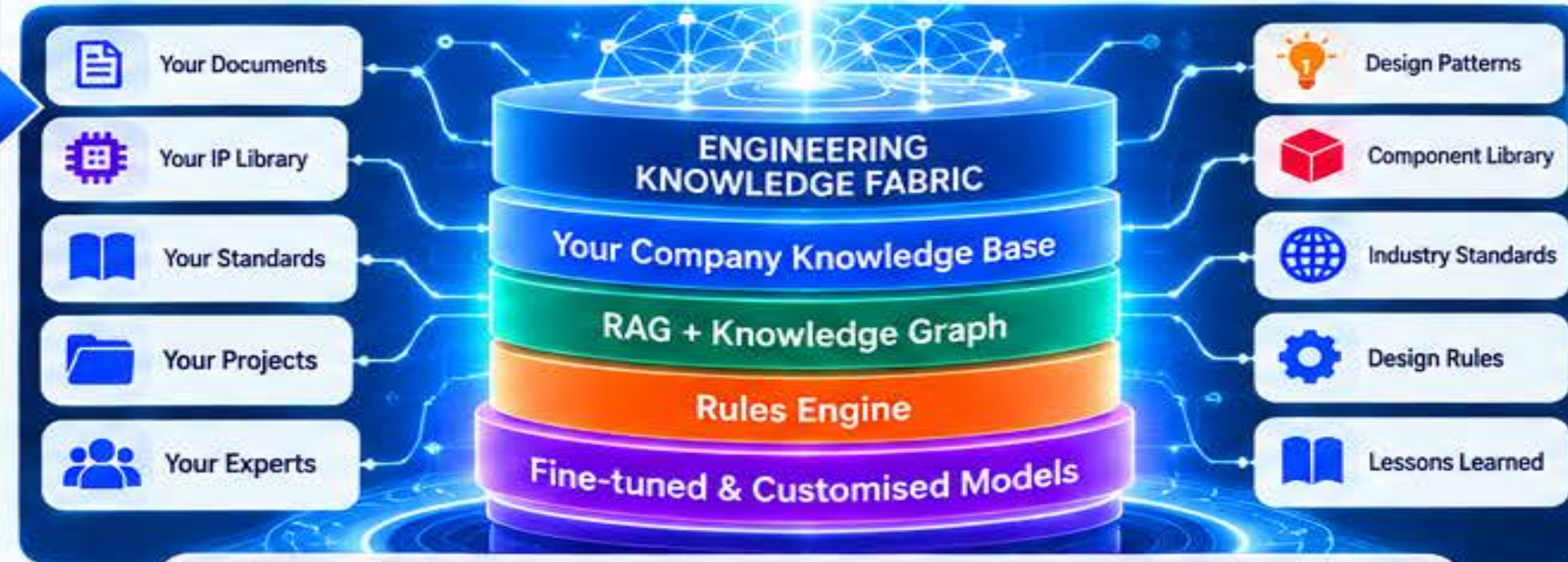


- Entities & relationships
- Projects, IP, components
- Design hierarchy
- Cross-domain links
- Traceability

Rules & Reasoning



- Company design rules
- Technology constraints
- Compliance checks
- Best practices
- Multi-agent reasoning



RAG + Knowledge Graph + Domain Models + Company Fine-tuning



Hybrid Search

- Keyword + Semantic
- Context aware
- Rank & relevance



Graph Reasoning

- Entity linking
- Cross-project learning
- Dependency analysis
- Traceability



Rules Engine

- Design & verification rules
- Constraint reasoning
- Compliance check
- Fact verification



Fine-tuned LLMs

- Your terminology
- Your design style
- Your standards
- Custom model training



ENGINEERING INTELLIGENCE

Your Knowledge. Your AI. Your Solutions. Continuously Improving.



- ✓ Company-aware
- ✓ Domain-adapted
- ✓ Higher accuracy
- ✓ Unique & Novel solutions

3 CUSTOMISED & PERSONALISED SOLUTIONS

AI agents leverage your knowledge to provide optimal solutions



Understands Your Domain

- Learns your terminology, design style and standards
- Uses your past projects and IPs for relevant solutions



Company-specific Recommendations

- Suggests architectures aligned with your methodology
- Recommends reusable IPs and components



Context-aware Solutions

- Considers your technology, constraints and design rules
- Provides trade-off analysis based on your data



Novelty & Innovation

- Combines your knowledge with latest industry advances
- Suggests novel architectures and optimizations



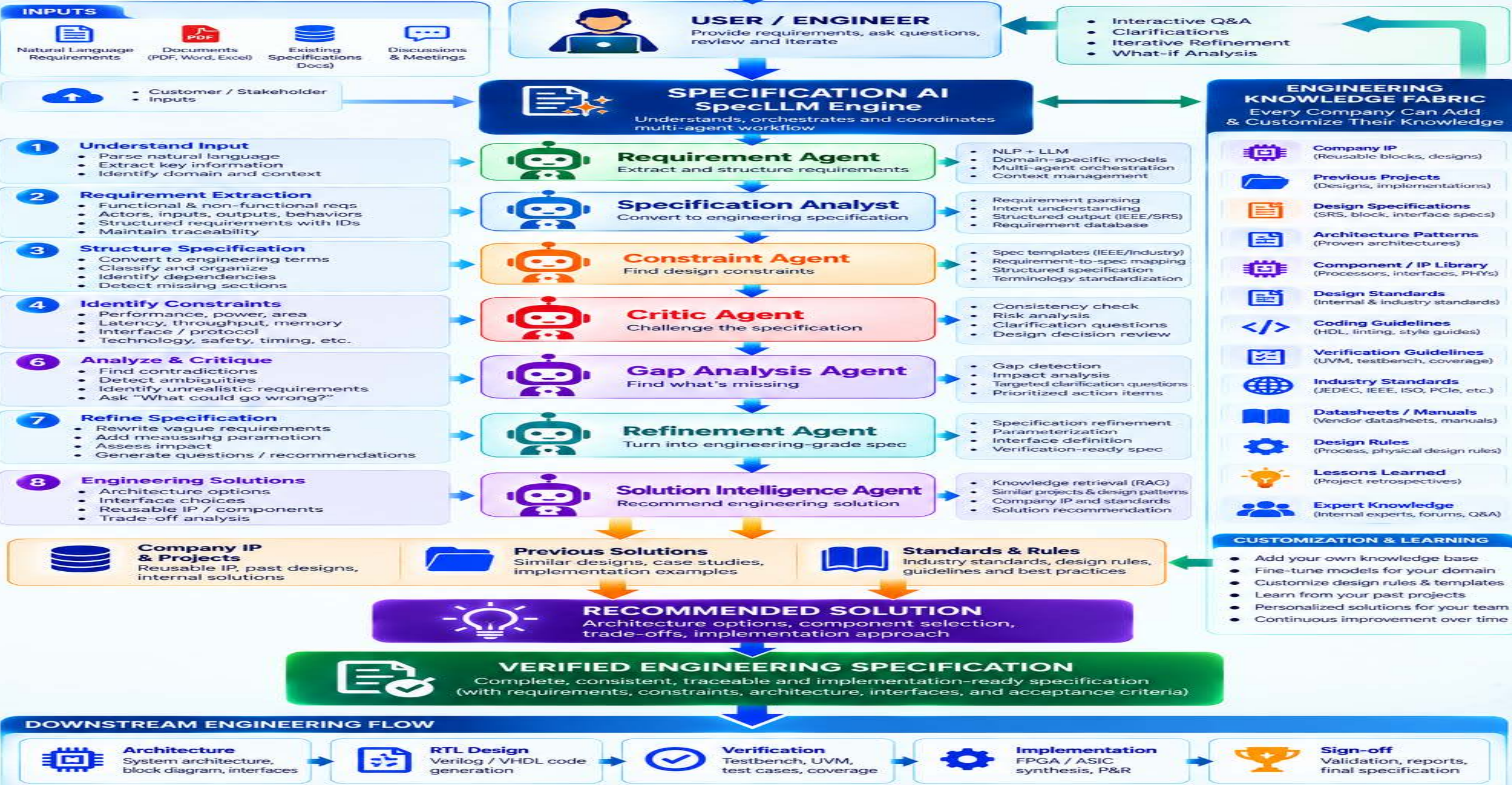
Personalised to Your Team

- Tailored to your use cases, preferences and goals
- Adapts over time with feedback and new projects

4 CONTINUOUS LEARNING & IMPROVEMENT

Your knowledge fabric gets smarter over time







NOVELTY & KEY DIFFERENTIATORS

#	Novelty	What makes it different
1	Multi-Agent Specification Intelligence	Multiple specialized agents collaborate instead of one generic LLM.
2	Requirement → Engineering Specification	Converts natural language into structured, measurable, traceable specifications.
3	AI Critic + Gap Analysis	Actively challenges requirements instead of simply accepting them.
4	Engineering Solution Intelligence	Recommends architectures, IP, interfaces and implementation approaches.
5	Company Knowledge Fabric	Each company can inject its own IP, projects, standards, rules and expert knowledge.
6	Company-Specific AI	RAG + Knowledge Graph + Rules + domain/customized models.
7	Continuous Learning	Learns from previous projects, design decisions, failures and feedback.
8	Traceability by Design	Requirement → Constraint → Decision → Architecture → RTL → Verification.
9	Human + AI Collaboration	Engineer remains in the loop for questions, approvals and design decisions.
10	Spec-to-Silicon Foundation	Specification becomes the starting point for Architecture, RTL, Verification and Sign-off.



HOW IT HELPS COMPANIES

From Better Specifications to Faster, Higher-Quality Silicon



Reduce Engineering Effort

- Automates repetitive tasks
- Faster documentation
- Less rework



Saved Time & Cost



Capture Institutional Knowledge

- Preserve expert knowledge
- Make it reusable
- Reduce dependency on individuals



Knowledge Retention



Reuse Previous Engineering Work

- Find similar projects
- Reuse IP, architectures, interfaces
- Leverage lessons learned



Faster Design Cycles



Improve Specification Quality

- Detect ambiguities and missing details
- Check feasibility and consistency
- Ensure completeness and traceability



Higher Quality Designs



Company-Specific Solutions

- Uses your IP, standards and design rules
- Tailored to your domain and processes
- Customizable and fine-tunable



Solutions Aligned to Your Process



Continuous Improvement

- Learns from past projects and feedback
- Better recommendations over time
- Builds your engineering intelligence



Growing Company AI



STRATEGIC VALUE FOR COMPANIES



Faster Time-to-Market



Higher Product Quality



Reuse IP & Reduce Cost



Build Long-Term Engineering Moat

YOUR ENGINEERING KNOWLEDGE. YOUR STANDARDS. YOUR SOLUTIONS. POWERED BY AI AGENTS.



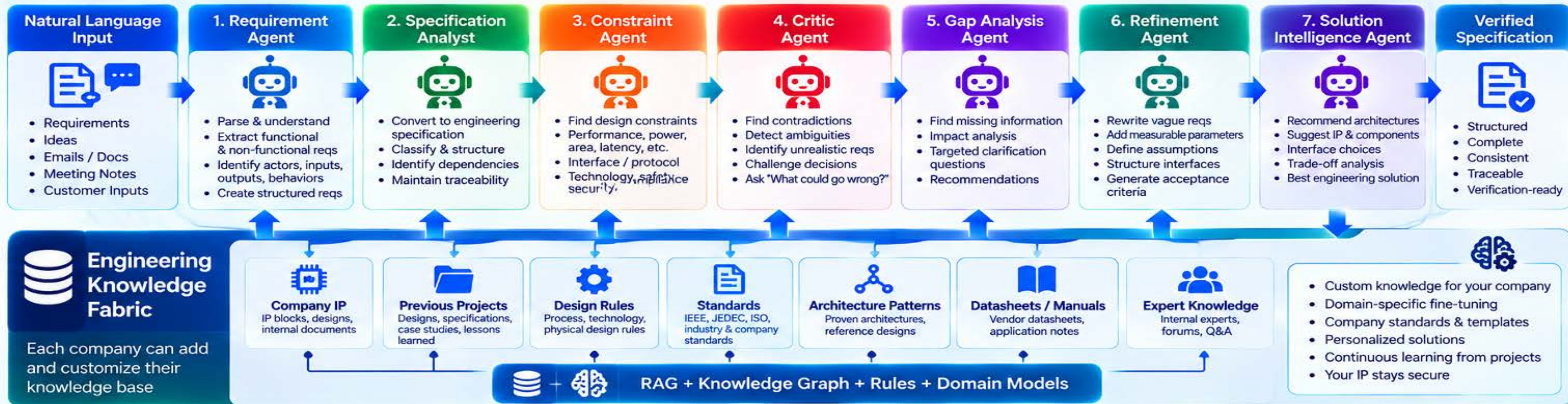
USP — Unique Selling Proposition

“Your Engineering Knowledge. Your Standards. Your Solutions. Powered by AI Agents.”



Core USP

An AI Engineering Specification Copilot that doesn't just write specifications— it **understands**, **challenges**, **improves**, **over company knowledge**, **recommends** engineering solutions, and produces **traceable**, **verification-ready** specifications.



HOW IT HELPS COMPANIES



Reduces Engineering Effort

- Automates repetitive tasks
- Faster documentation
- Less rework



Captures Institutional Knowledge

- Preserve expert knowledge
- Make it reusable
- Reduce dependency on individuals



Reuses Previous Work

- Find similar projects
- Resue IP and architectures
- Leverage lessons learned



Improves Specification Quality

- Detect missing/ambiguous details
- Check feasibility & consistency
- Ensure traceability



Faster Design Cycles

- From weeks to days
- AI-driven analysis and recommendations
- Focus on critical decisions



Builds a Long-Term Moat

- More projects
- More knowledge
- Better AI recommendations
- Better designs



Specification to Silicon

Verified
Specification

Architecture
(System Design)

RTL Design
(Verilog / VHDL)

Verification
(Testbench, UVM)

Implementation
(ASIC / FPGA)

Sign-off
(Validation, Reports)

HOW IT HELPS COMPANIES

From Better Specifications to Faster, Higher-Quality Designs and Silicon

A Reduce Engineering Effort

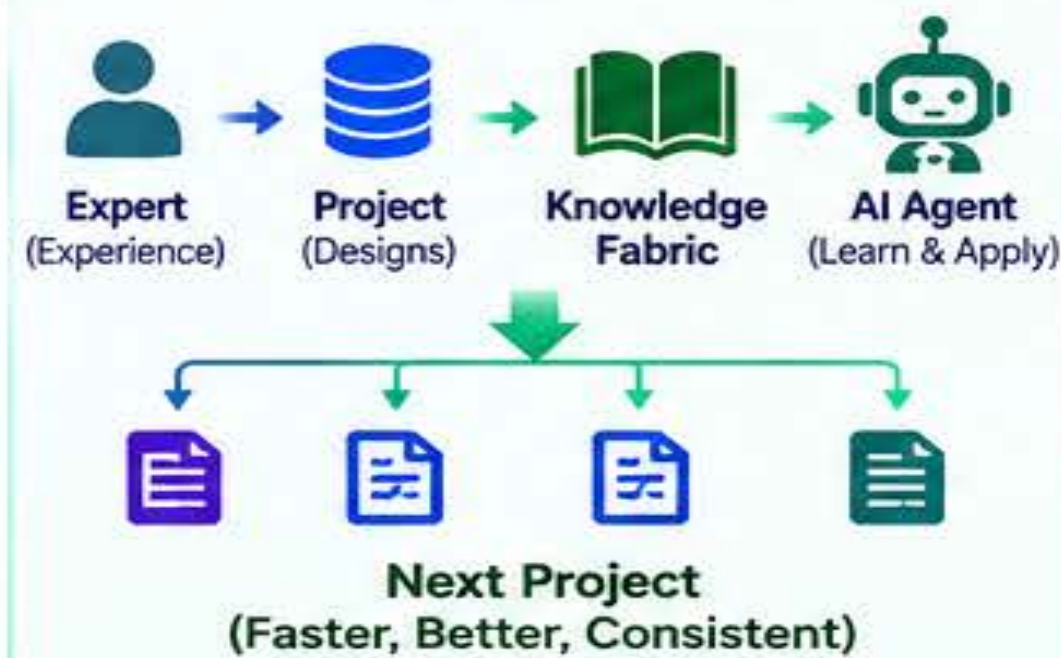
Automates repetitive specification activities

- Requirement extraction
- Documentation
- Classification
- Gap identification
- Specification formatting
- Traceability
- Review preparation

Save Time & Cost
Let engineers focus on high-value engineering and innovation.

B Capture Institutional Knowledge

Company knowledge becomes reusable AI intelligence.



- ✓ Preserve expert knowledge
- ✓ Reduce dependency on individuals
- ✓ Build a long-term knowledge asset
- ✓ Capture lessons learned
- ✓ Continuous learning from new projects

C Reuse Previous Engineering Work

Discover and leverage existing knowledge and assets.

- Similar previous projects
- Existing IP
- Previous architectures
- Proven interfaces
- Previous specifications
- Lessons learned
- Known design problems

Don't reinvent the wheel
Reuse what already works and build better solutions faster.

D Improve Specification Quality

AI continuously checks and enhances specifications.

- Completeness**
(All required information)
- Consistency**
(No contradictions)
- Constraints**
(Performance, power, area, etc.)
- Ambiguity**
(Identify unclear statements)
- Feasibility**
(Realistic and implementable)
- Traceability**
(End-to-end traceability)

Higher Quality Specifications
Reduce costly errors in RTL, verification and implementation.

E Faster Design Cycles

From weeks/months to days with AI-powered analysis and recommendations.

Traditional Flow (Slow)



Weeks / Months

With SpecificationAI (Fast)



Days / Weeks

BUSINESS IMPACT
Higher Productivity.
Better Designs.
Faster to Market.

Save Engineering Time

50-70%
reduction in specification effort

Leverage Existing Knowledge

Reuse
IP, architectures and proven solutions

Improve Quality

Fewer
errors in RTL, verification and implementation

Faster Time-to-Market

Reduce
design cycle from months to days/weeks

Build Long-Term Value

Continuous
improvement with every new project



Your Engineering Knowledge



Your Standards



Your Solutions



Powered by AI Agents



Higher Quality Designs



Long-Term Competitive Advantage

STRATEGIC COMPANY VALUE

Turn Individual Knowledge into a Compounding Company Asset

TRADITIONAL ENGINEERING KNOWLEDGE

Knowledge stays with individuals and is easily lost



Repeated effort, reinvention and loss of institutional knowledge

WITH SPECIFICATION AI

Knowledge becomes a reusable company asset



Knowledge preserved, reusable and continuously growing

THE LONG-TERM MOAT

Every project makes the company's AI more valuable



BUSINESS IMPACT



Preserve institutional knowledge
Reduces dependency on individuals



Higher engineering productivity
Reuse and faster execution



Better quality designs
Fewer errors, less rework



Continuous improvement
Every project makes the AI smarter



Long-term competitive advantage
Unique company-specific engineering intelligence

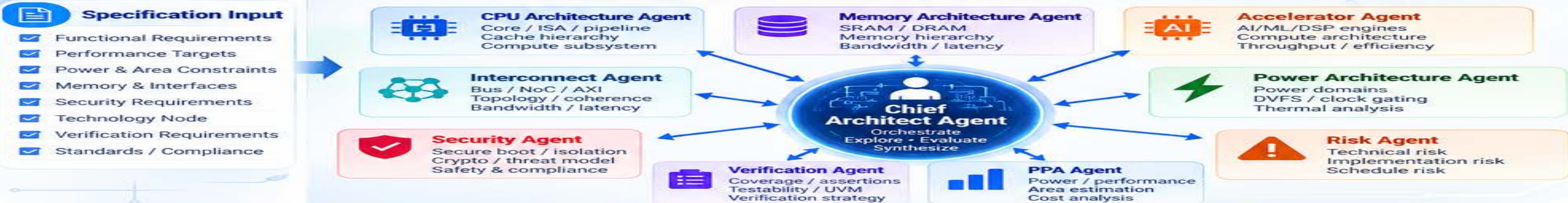
SiliconAI Architect: Interactive Specification → Architecture

“An AI-powered interactive architecture co-pilot that transforms engineering specifications into optimized chip architectures through multi-agent design exploration, enterprise knowledge, PPA analysis, and human-in-the-loop architectural decisions.”

Specification Intelligence + Engineering Knowledge + Multi-Agent Architecture Exploration + Interactive Architect Collaboration + PPA/Risk Reasoning + Human Approval + Reusable Design Knowledge.

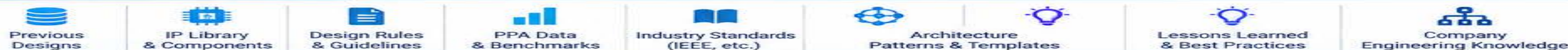


Multi-Agent Architecture Exploration



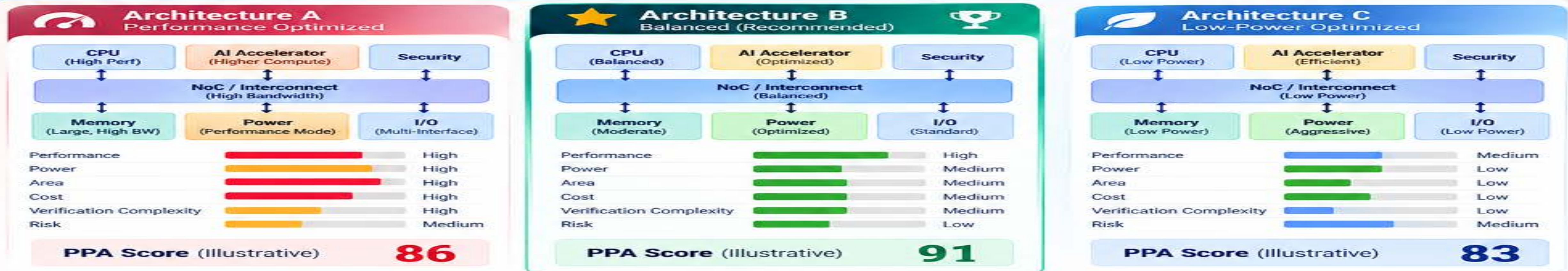
Knowledge Base

Retrieve and apply organizational engineering knowledge



Candidate Architecture Comparison

Illustrative values for comparison (not actual measurements)



Architect Decision Loop (Human-in-the-Loop)

Interactive, explainable and iterative architecture design



Key Benefits





Specification is the starting point and drives the entire process



Key Benefits



Optimized Architectures



Faster Design Exploration



Knowledge-Driven Decisions



Human-in-the-Loop for Trust & Control



PPA, Cost & Risk Analysis



Reduced Time-to-Market



Reusable Engineering Intelligence

Multi-Agent Architecture Exploration

Specialized AI agents collaboratively explore, analyze and optimize chip architecture alternatives



CPU Agent

Core / ISA / cache



Memory Agent

Hierarchy / bandwidth



Accelerator Agent

Compute architecture



Interconnect Agent

Bus / NoC / coherence



Power Agent

Power domains / DVFS



Security Agent

Threat model / isolation



Verification Agent

Coverage / testability



PPA Agent

Power / Performance / Area



Risk Agent

Technical / implementation risk



Candidate Architecture Comparison



Architecture A

Performance optimized



PPA Score: **86**

Review A >



Architecture B

Balanced (Recommended)



PPA Score: **91**

Review B >



Architecture C

Low-power optimized



PPA Score: **83**

Review C >



Human-in-the-Loop Decision Gate

AI recommendations remain proposals until the architect approves or modifies them.



Architect Reviews AI Recommendations



Approve Recommendation

Select this architecture as the baseline.



Modify Architecture

Update constraints, refine and iterate.



Reject & Explore Again

Provide feedback and generate new alternatives.

Iterate with new insights and constraints



Architecture Knowledge Capture

Every approved decision can become reusable organizational engineering intelligence.



Decision Record

What was decided, when and why.



Design Rationale

Why this architecture was chosen.



PPA Evidence

Performance, Power, Area analysis.



Architect Feedback

Expert inputs, comments and reviews.



Reusable Pattern

Convert to design pattern or template.



Lessons Learned

Capture insights and best practices.



Approved Architecture

- ✓ Design Decisions
- ✓ PPA Data
- ✓ Design Rationale
- ✓ Architect Feedback

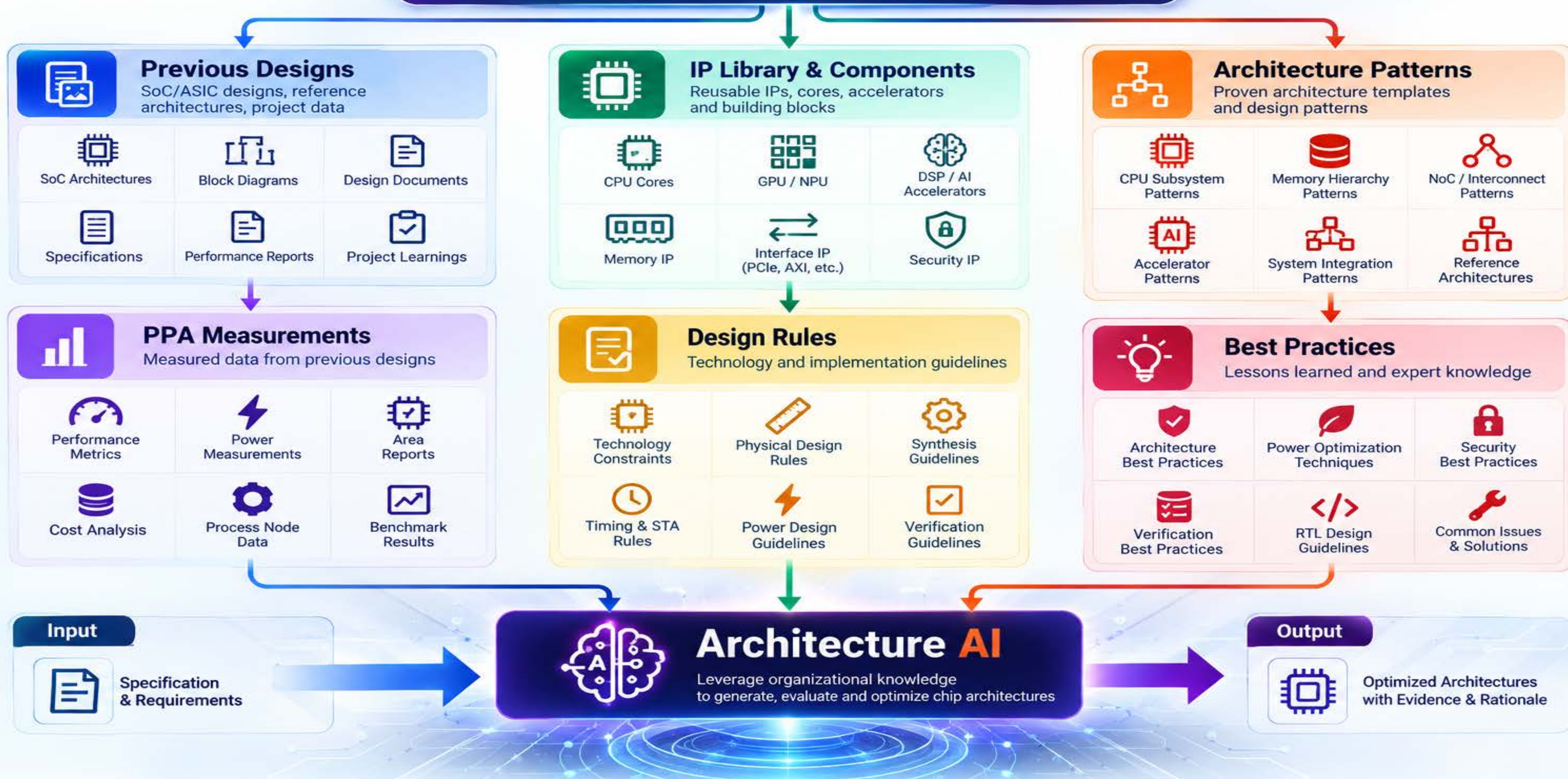


Organizational Engineering Intelligence (Knowledge Base)

- ✓ Reusable Patterns
- ✓ Templates & Guidelines
- ✓ Lessons Learned
- ✓ Best Practices



Future Architectures

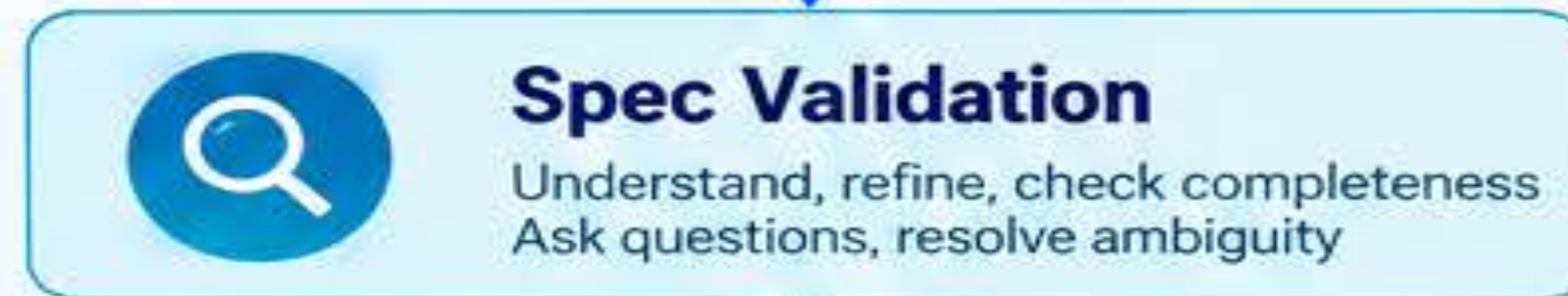


Let me validate the specification and identify missing details...



Input from Architect

- Functional & performance requirements
- Power, area and cost constraints
- Memory and interface requirements
- Security and verification requirements
- Workloads and use cases



AI Analysis

- ✓ Identify missing requirements
- ✓ Check for conflicts and feasibility
- ✓ Ask clarifying questions
- ✓ Generate refined requirement model



Architect Review

- Answer AI questions
- Resolve ambiguities
- Approve requirement baseline



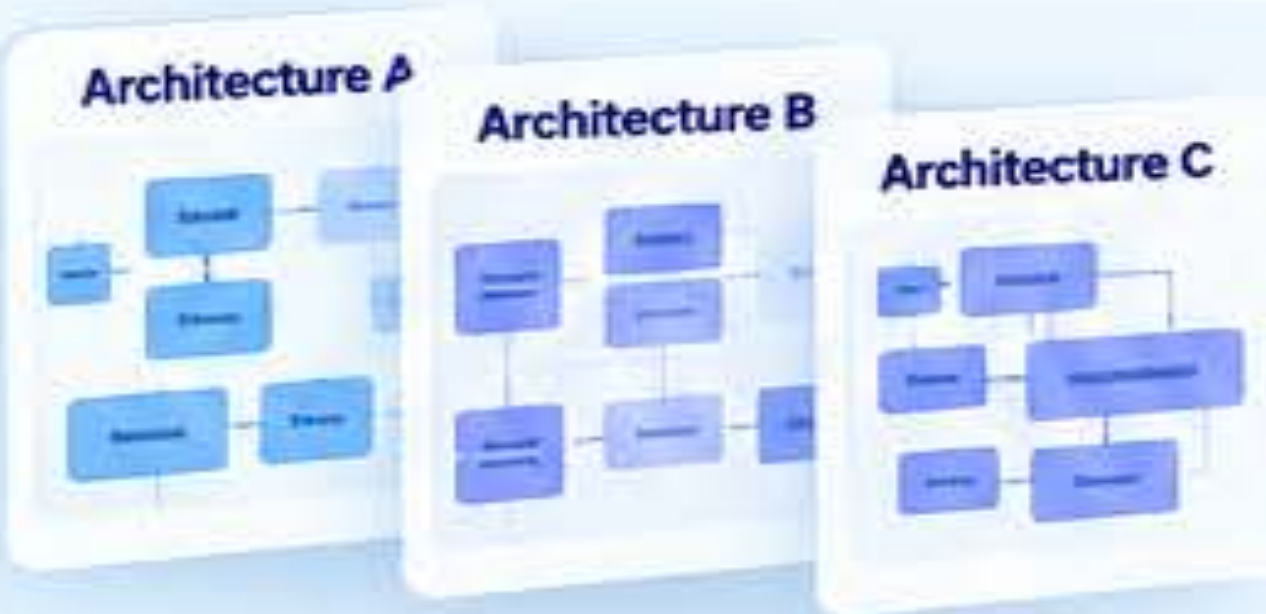
Candidate Architectures

- Performance optimized
- Balanced (recommended)
- Low-power optimized



Architect Decision

- Select architectures to analyze
- Provide additional constraints
- Request new alternatives if needed





Multi-Agent Analysis

Specialized AI agents analyze, evaluate and compare selected architectures



CPU Agent



Memory Agent



Accelerator Agent



Interconnect Agent



Power Agent



Security Agent



Verification Agent



PPA Agent



Risk Agent

Analysis Outputs

- PPA (Power, Performance, Area)
- Cost analysis
- Risk assessment
- Verification complexity
- Trade-off comparison
- Recommendations with rationale



HUMAN GATE #3

Review Trade-offs

Evaluate analysis results and recommendations

Architect Review

- Compare alternatives
- Understand trade-offs (Why?)
- Provide feedback and modify constraints



Architecture Baseline

Selected architecture with key decisions, constraints and rationale

Baseline Deliverables

- Block diagram
- Key architectural parameters
- Interface definitions
- Design constraints
- Assumptions and rationale



HUMAN GATE #4

Architect Approval

Approve architecture baseline for detailed design

Final Approval

- Review complete baseline
- Confirm alignment with goals
- Approve to proceed to detailed architecture



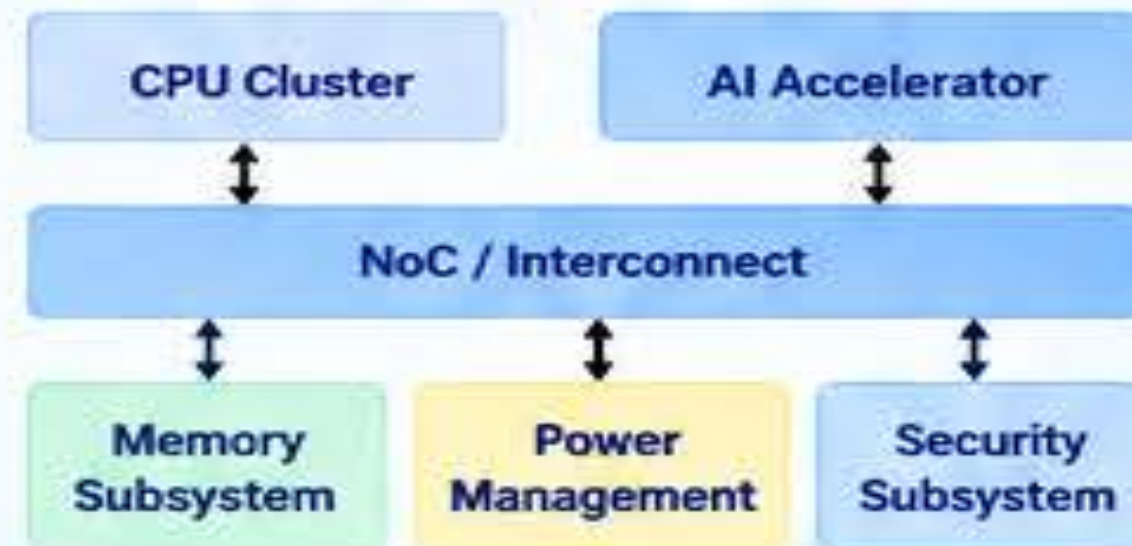
Detailed Architecture

Comprehensive architecture specification for implementation (RTL, verification, etc.)

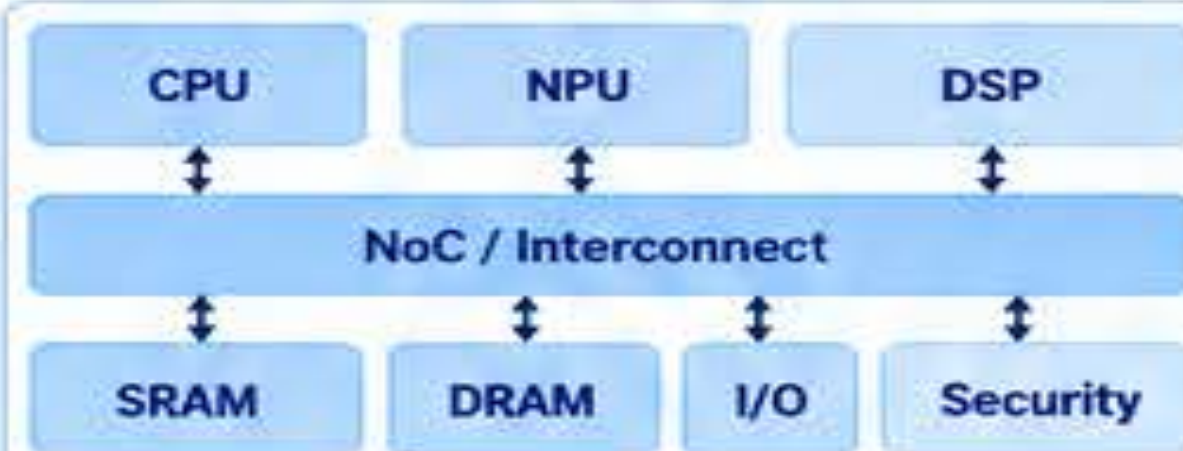
Detailed Outputs

- Complete architecture specification
- Detailed block diagrams
- Interfaces and protocols
- Configuration and parameters
- Ready for RTL and verification

Example Baseline Architecture



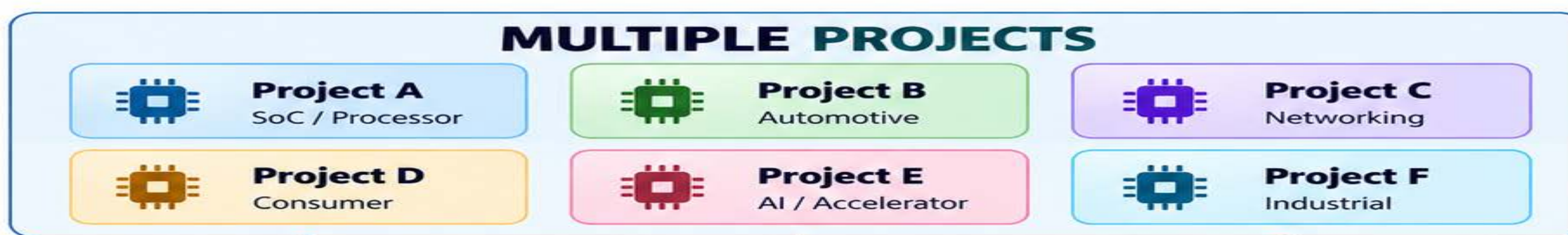
Detailed Architecture (Example)



SiliconAI IP — AI-Powered IP Intelligence & Reuse

Discover. Understand. Evaluate. Reuse. Integrate.

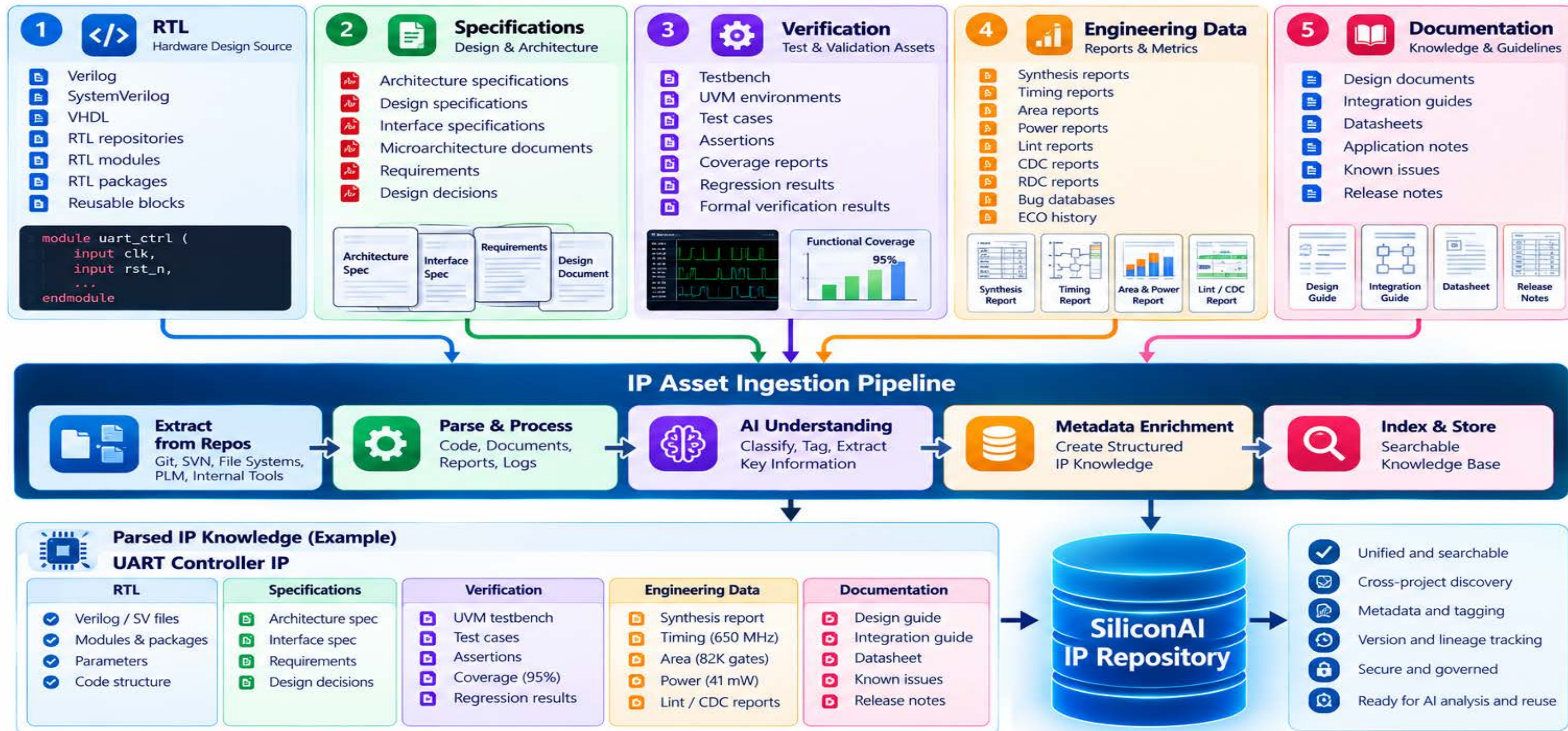
Before you build a new IP, SiliconAI tells you what your organization already knows, what it already owns, what can be reused, what must be modified, and what should be built from scratch.

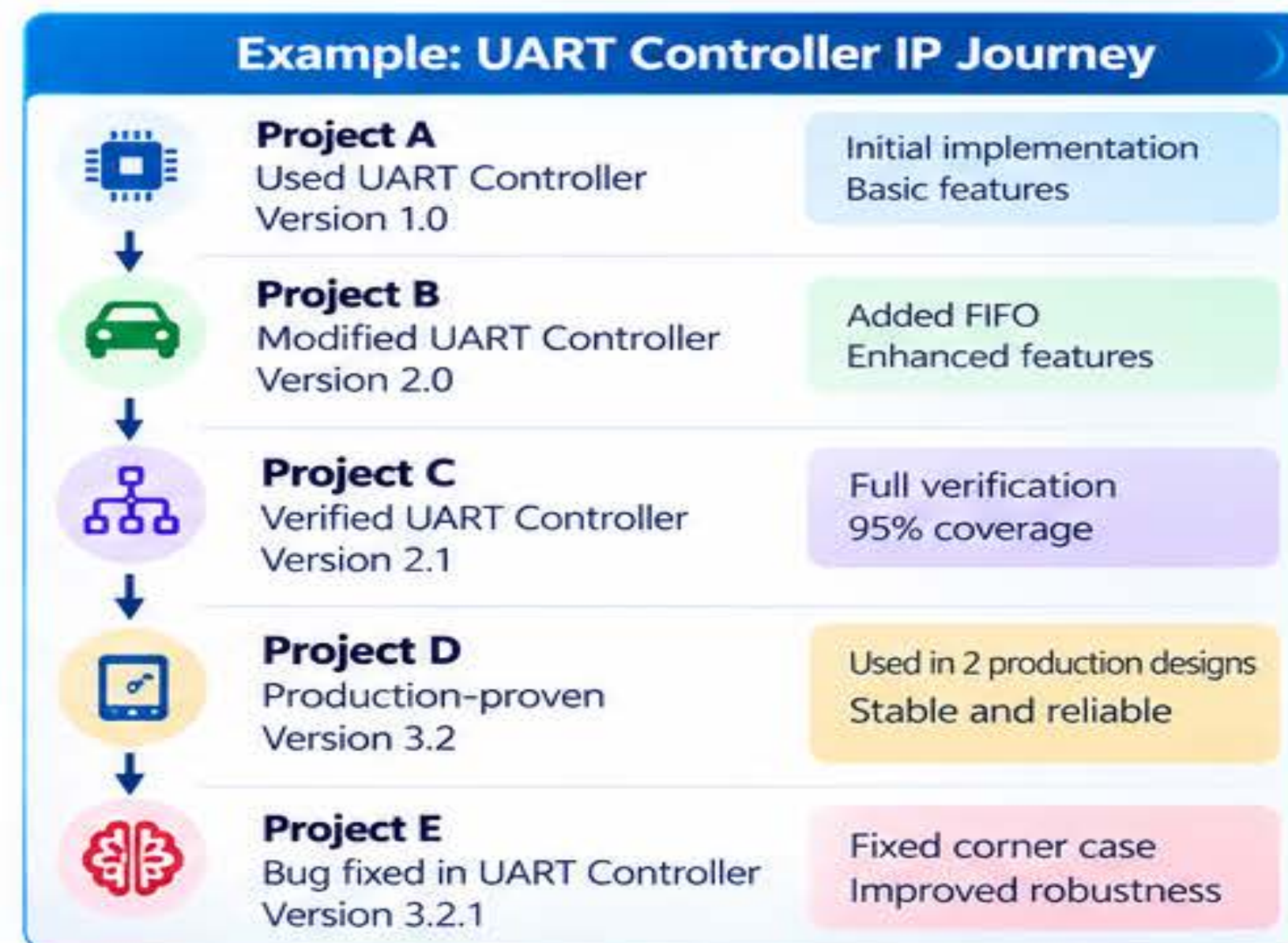
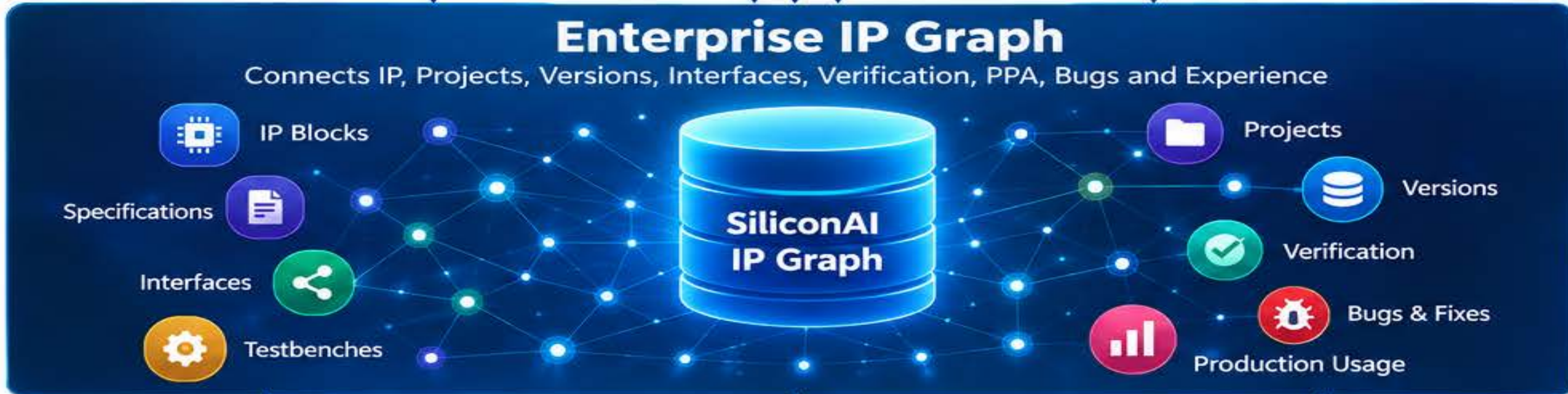
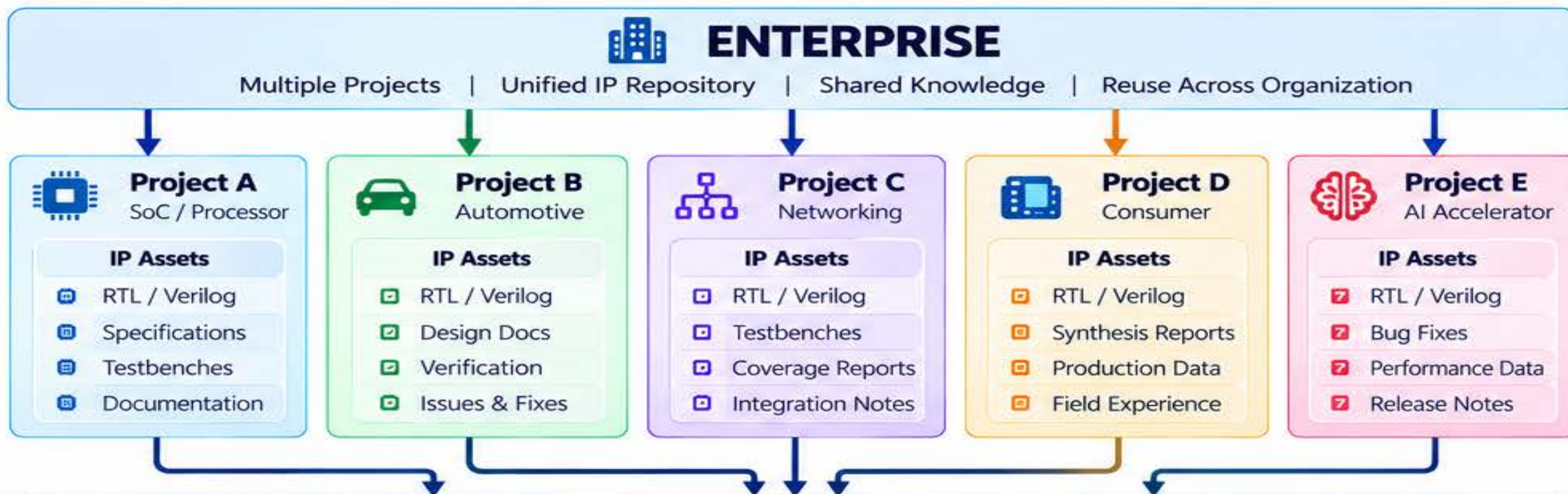


SiliconAI IP IP Asset Ingestion

Ingest, Parse and Understand All Types of Engineering Assets Across Projects

Bring All Your Engineering Knowledge into a Unified IP Repository
From RTL to Reports, Documentation to Verification





Comparison of Versions					
Version	Source Project	Key Features	Verification Coverage	Status	Recommendation
v1.0	A	Basic UART	78%	Prototype	-
v2.0	B	+ FIFO, Enhanced	85%	Internal Use	-
v2.1	C	Verified, Stable	95%	Validated	-
v3.2	D	Production Ready	97%	Production	RECOMMENDED
v3.2.1	E	Bug Fixes	97%	Stable	Alternative



Traditional Search

Search: **DMA**



- ✗ Keyword based search
- ✗ Limited results
- ✗ Misses similar or renamed IP
- ✗ No technical evaluation
- ✗ No compatibility analysis



SiliconAI Intelligent Search

Natural Language Requirement:

“Need a high-throughput memory-to-memory data movement engine supporting AXI4, programmable burst sizes and multiple outstanding transactions.”



- ✓ Understands intent and context
- ✓ Searches across all projects and documents
- ✓ Finds similar and related IP
- ✓ Evaluates technical compatibility
- ✓ Provides ranked results with recommendations



AI Understanding & Search Dimensions



Semantic Meaning

Understand intent, domain and engineering context

+



Functional Behavior

Identify key functions and capabilities

+



Interface Requirements

AXI4, AHB, APB, custom interfaces

+



Protocol

AXI4, AXI3, AHB, PCIe, etc.

+



Performance

Throughput, latency, burst size, outstanding transactions

+



Architecture

Microarchitecture, data path, control

+



Technology

Process node, technology, libraries

+



Verification

Coverage, regression, signoff status



Intelligent IP Discovery Results

Ranked candidates based on multi-dimensional analysis

Rank	Candidate IP	Match Score	Status	Recommendation
1	DMA-IP-07 High Performance DMA with AXI4	96%	Production	Reuse
2	DMA-IP-12 Multi-channel DMA AXI4, Scatter-Gather	91%	Production	Reuse
3	DMA-IP-03 Configurable DMA AXI4, Single Channel	82%	Prototype	Modify
4	DMA-IP-19 Legacy DMA AXI3	71%	Experimental	Avoid
–	No suitable IP found	–	–	Build New

Top Candidate Details (DMA-IP-07)



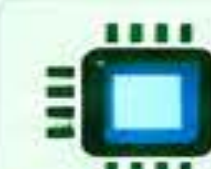
- ✓ High-throughput memory-to-memory data movement
- ✓ AXI4 interface, programmable burst sizes
- ✓ Multiple outstanding transactions
- ✓ Scatter-Gather support
- ✓ Error handling and interrupts
- ✓ Production proven in multiple projects

Specification	Details
Interface	AXI4
Max Throughput	32 GB/s
Data Width	128 / 256 bit
Burst Size	Programmable (up to 256)
Outstanding Txn	Up to 16
Architecture	Multi-channel, Scatter-Gather
Technology	7nm / 12nm
Verification	98% functional coverage
Usage	2 production projects

Comparison of Top Candidates

Feature	DMA-IP-07	DMA-IP-12	DMA-IP-03	DMA-IP-19
Match Score	96%	91%	82%	71%
Status	Production	Production	Prototype	Experimental
Interface	AXI4	AXI4	AXI4	AXI3
Data Width	256 bit	128 bit	128 bit	64 bit
Max Throughput	32 GB/s	24 GB/s	12 GB/s	4 GB/s
Burst Size	Up to 256	Up to 128	Up to 64	Up to 16
Outstanding Txn	16	8	4	1
Verification	98%	95%	82%	60%
Technology	7nm	12nm	28nm	40nm
Integration Effort	Low	Low	Medium	High

AI Recommendation



Use DMA-IP-07

It has the highest match score (96%), production proven, and already used in two production designs.

- ✓ Meets all functional requirements
- ✓ AXI4 interface compatible
- ✓ Supports high throughput and burst sizes
- ✓ Multiple outstanding transactions
- ✓ High verification maturity (98%)
- ✓ Low integration effort
- ✓ Reduced development time and risk

SiliconAI IP – IP Knowledge Extraction

From Documents to Engineering Intelligence

Turn Engineering Assets into **Actionable IP Knowledge**

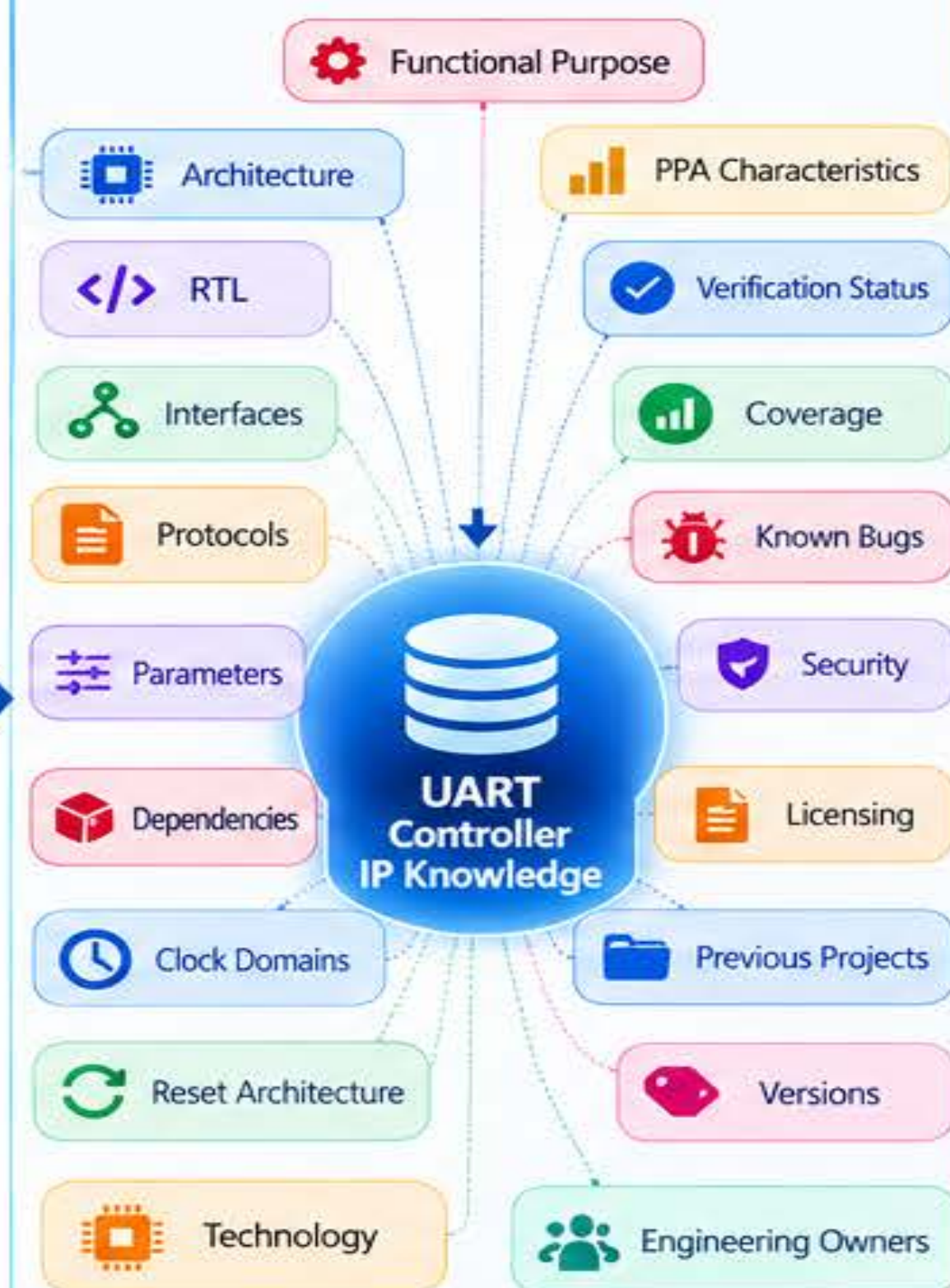
Engineering Assets (Multiple Formats & Sources)

- RTL**
 - Verilog / SystemVerilog
 - VHDL
 - RTL repositories
 - RTL modules / packages
 - Reusable blocks
- Specifications**
 - Architecture specifications
 - Design specifications
 - Interface specifications
 - Microarchitecture documents
 - Requirements
 - Design decisions
- Verification**
 - Testbench
 - UVM environments
 - Test cases
 - Assertions
 - Coverage reports
 - Regression results
 - Formal verification results
- Engineering Data**
 - Synthesis reports
 - Timing reports
 - Area reports
 - Power reports
 - Lint reports
 - CDC / RDC reports
 - Bug databases
 - ECO history
- Documentation**
 - Design documents
 - Integration guides
 - Datasheets
 - Application notes
 - Known issues
 - Release notes

AI-Powered Extraction & Understanding

- Ingest & Parse**
Extract content from code, documents, reports and logs
- AI Understanding**
LLM + domain models understand meaning and context
- Information Extraction**
Extract key entities, relationships and engineering facts
- Normalize & Structure**
Standardize and create consistent IP metadata
- Build IP Knowledge Graph**
Link across documents, projects, versions and dependencies
- Generate Knowledge Card**
Create structured, searchable engineering IP profile

Structured Engineering Knowledge



From Documents to Engineering Knowledge

- Unstructured Files**
 - PDF, DOC, XLS, RTL, Logs
 - Reports, presentations
 - Scattered across projects
- AI Knowledge Extraction**
 - Understand, extract, structure
 - Link related information
 - Create searchable knowledge
- Structured IP Knowledge**
 - Rich metadata and insights
 - Searchable and comparable
 - Actionable for reuse and design

IP Knowledge Card – UART Controller

IP Name UART Controller	IP ID UART-IP-07	Production Ready
Category Communication IP	Version v3.2	

Functional Purpose Asynchronous serial communication controller supporting transmit and receive with configurable baud rate, FIFO and interrupt support.	Architecture - Configurable, modular architecture - TX/RX with FIFO - Clock domain independent - Optional parity/ECC - Low power modes
RTL - SystemVerilog - Clean, synthesizable RTL - Parameterized design - UVM verified	Interfaces - AXI4-Lite (control) - APB (optional) - UART TX/RX pins - Interrupt interface
Protocols - UART (standard) - AXI4, APB - Custom handshake (optional)	Parameters - Data width: 8/16/32 bit - FIFO depth: 16/64/256 - Baud rate: 9600 – 3 Mbps - Parity: None/Even/Odd
Dependencies - AXI4-Lite interconnect - Clock and reset infrastructure - Optional interrupt controller	Clock Domains - Core clock: 50–200 MHz - Asynchronous UART clock - CDC for TX/RX
Reset Architecture - Active low reset (configurable) - Synchronous reset - Separate reset for core and FIFO	Technology - Process: 7nm / 12nm / 28nm - Standard cell library - FPGA compatible
PPA Characteristics - Area: 12K gates - Power: 2.5 mW @ 100 MHz - Frequency: 500 MHz (max)	Verification Status - UVM environment - Regression complete - Signoff: Yes
Coverage - Functional coverage: 98% - Code coverage: 96% - Assertion coverage: 92%	Known Bugs #102: Baud rate glitch (fixed in v3.1) #145: FIFO overflow in rare case (fixed) - No open critical bugs
Security - Access control on registers - No known security vulnerabilities - Security scan: Passed	Licensing - Internal IP (Company owned) - Reuse allowed within organization - Export restrictions: No
Previous Projects - Project A (v2.1) – Prototype - Project B (v3.0) – Production - Project C (v3.2) – Production	Versions - v1.0 – Initial - v2.1 – Added FIFO - v3.0 – CDC improvement - v3.2 – Production (current)
Engineering Owners - Design Owner: ABC Team - Verification Owner: DEF Team - Primary Contact: XYZ - Last Updated: Sep 2026	

1 Functional Compatibility

Does the IP actually satisfy the requirement?

- Requirement**
Understand and extract key functions
- Functional Decomposition**
Break into features and sub-features
- IP Capabilities**
Extract IP features and capabilities
- Feature-by-Feature Mapping**
Compare and evaluate match

Example: DMA Engine Requirement

“Need a high-throughput memory-to-memory data movement engine supporting AXI4, programmable burst sizes and multiple outstanding transactions.”

Requirement	IP Capability	Match
AXI4 interface	Supported	✓
256-bit data width	Supported	✓
4K burst size	Supported	✓
8 outstanding requests	Supported	✓
ECC support	Not supported	✗
Low-power mode	Not supported	✗

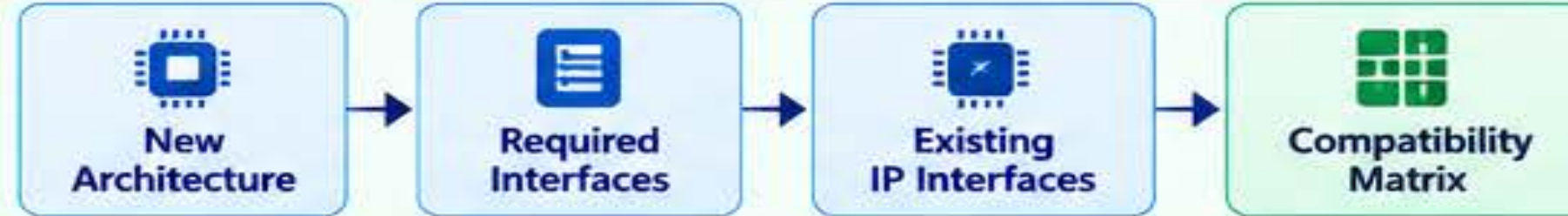


AI Conclusion

- 78% functional compatibility
- Core functionality matches
- Missing features: ECC, Low-power mode
- Recommended:** Reuse with modification

2 Interface Compatibility

Analyze interfaces, protocols and connectivity



Interface Comparison

Interface / Protocol	Required	IP Support	Compatibility
AXI4	✓ Yes	✓ Yes	Compatible
AHB	✗ No	✗ No	N/A
APB	✓ Yes	✗ No	Adapter Required
ACE	✗ No	✗ No	N/A
CHI	✗ No	✗ No	N/A
PCIe	✗ No	✗ No	N/A
USB	✗ No	✗ No	N/A
Ethernet	✗ No	✗ No	N/A
SPI	✗ No	✗ No	N/A
I²C	✗ No	✗ No	N/A
MIPI	✗ No	✗ No	N/A
Custom Interface	Optional	Partial	With Adapter

Interface Analysis Results

- ✓ AXI4 interface fully compatible
- ✓ APB interface not available – requires AXI to APB adapter
- ✓ Other interfaces not required
- ✓ Custom interface partially compatible – minor adaptation

Identified Mismatches

- Signal mismatch**
APB interface not available
- Width mismatch**
AXI 128-bit → 256-bit (adapter)
- Protocol mismatch**
Custom interface timing
- Reset mismatch**
Active-low vs active-high
- Endianness mismatch**
Little-endian vs Big-endian
- Latency mismatch**
Additional latency in adapter

Recommendation

Adapter Required

AXI to APB bridge and width adapter needed.
Integration is feasible with standard adapters.

3 Clock & Reset Compatibility

Analyze clock domains, reset domains and CDC/RDC

Clock Domain Analysis



Reset Architecture Analysis



CDC / RDC Analysis

Clock Domain Crossing (CDC)	Not required (Same 250 MHz domain)	✓
Reset Domain Crossing (RDC)	Required (Async reset to sync reset)	⚠
Asynchronous Boundaries	None	✓
Reset Synchronization	Required (2-flop sync)	⚠
CDC Bridge Needed	Not required	✓
RDC Bridge Needed	Required	⚠

Recommendation

No Clock Adapter Required

- IP runs in existing 250 MHz domain
- Add reset synchronizer for RDC

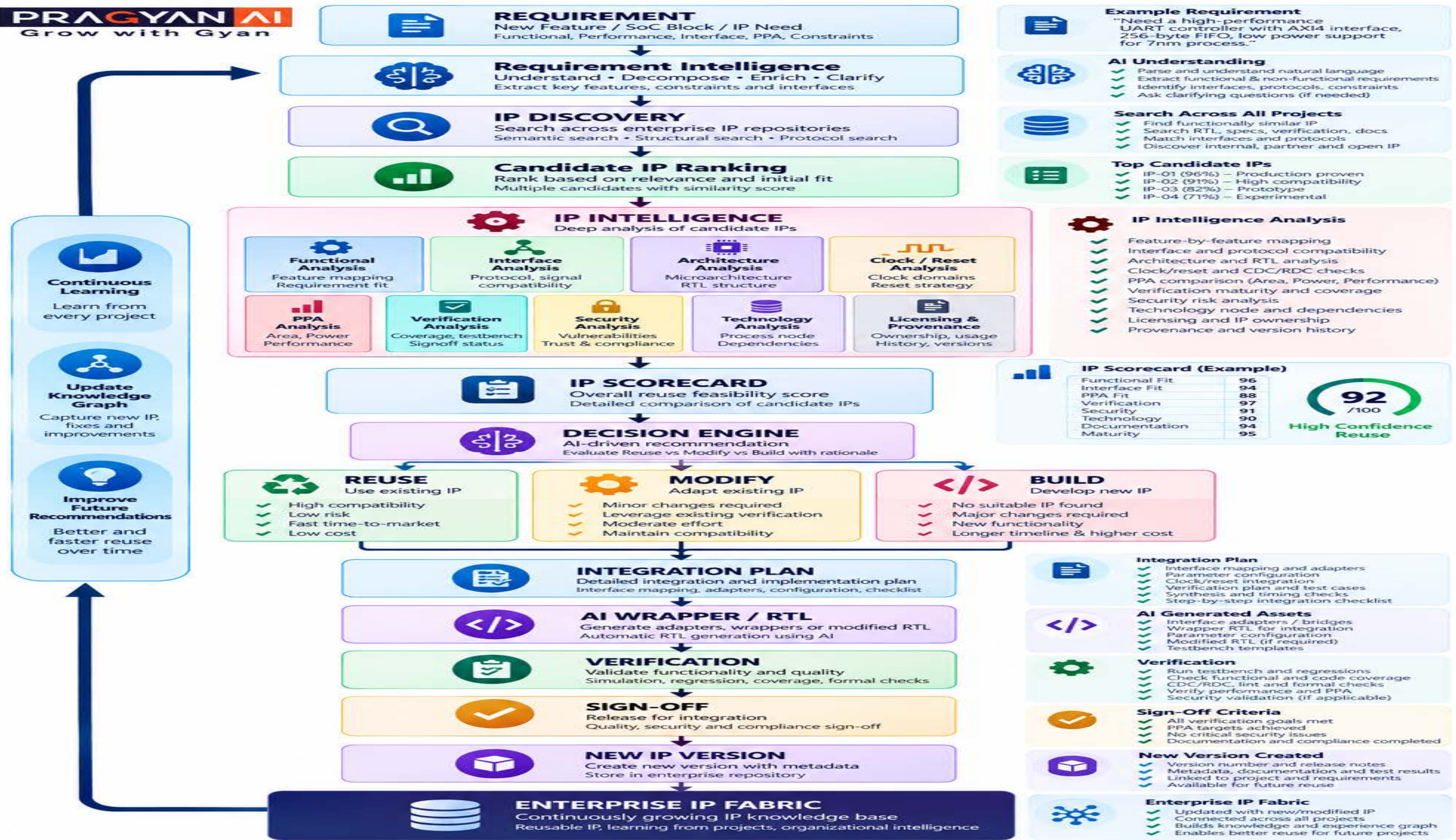
Overall Compatibility Summary

Multi-dimensional analysis for reuse decision



- Integration Feasibility**
Core functionality matches the requirement.
- ✓ Interface adapters needed (AXI to APB, width adapter).
- ✓ Clock domain is compatible (no adapter required).
- ✓ Reset synchronization required.
- ✓ Integration is feasible with moderate engineering effort.

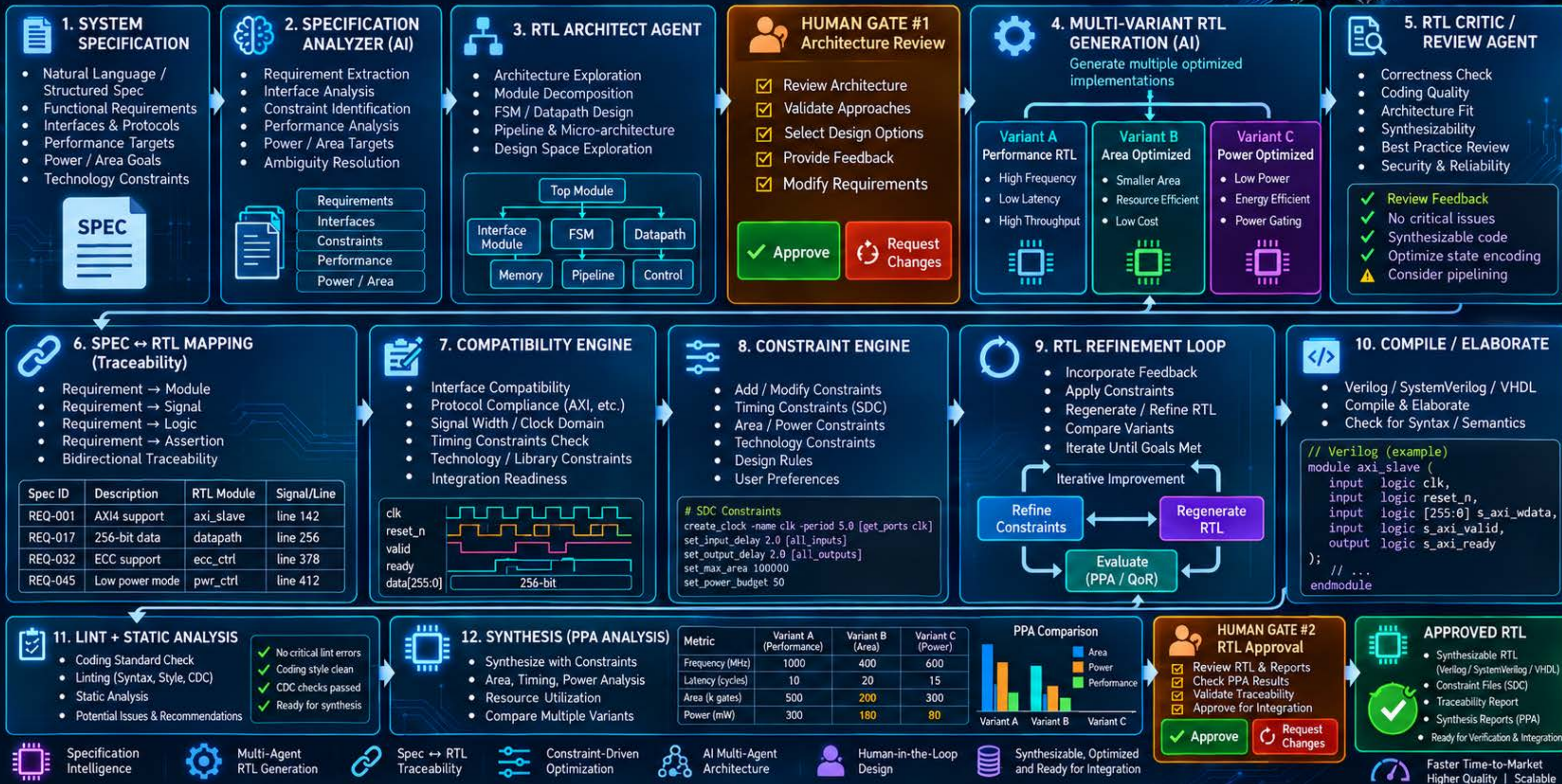
AI Recommendation
REUSE with Modification
Estimated Integration Effort
2 – 3 weeks



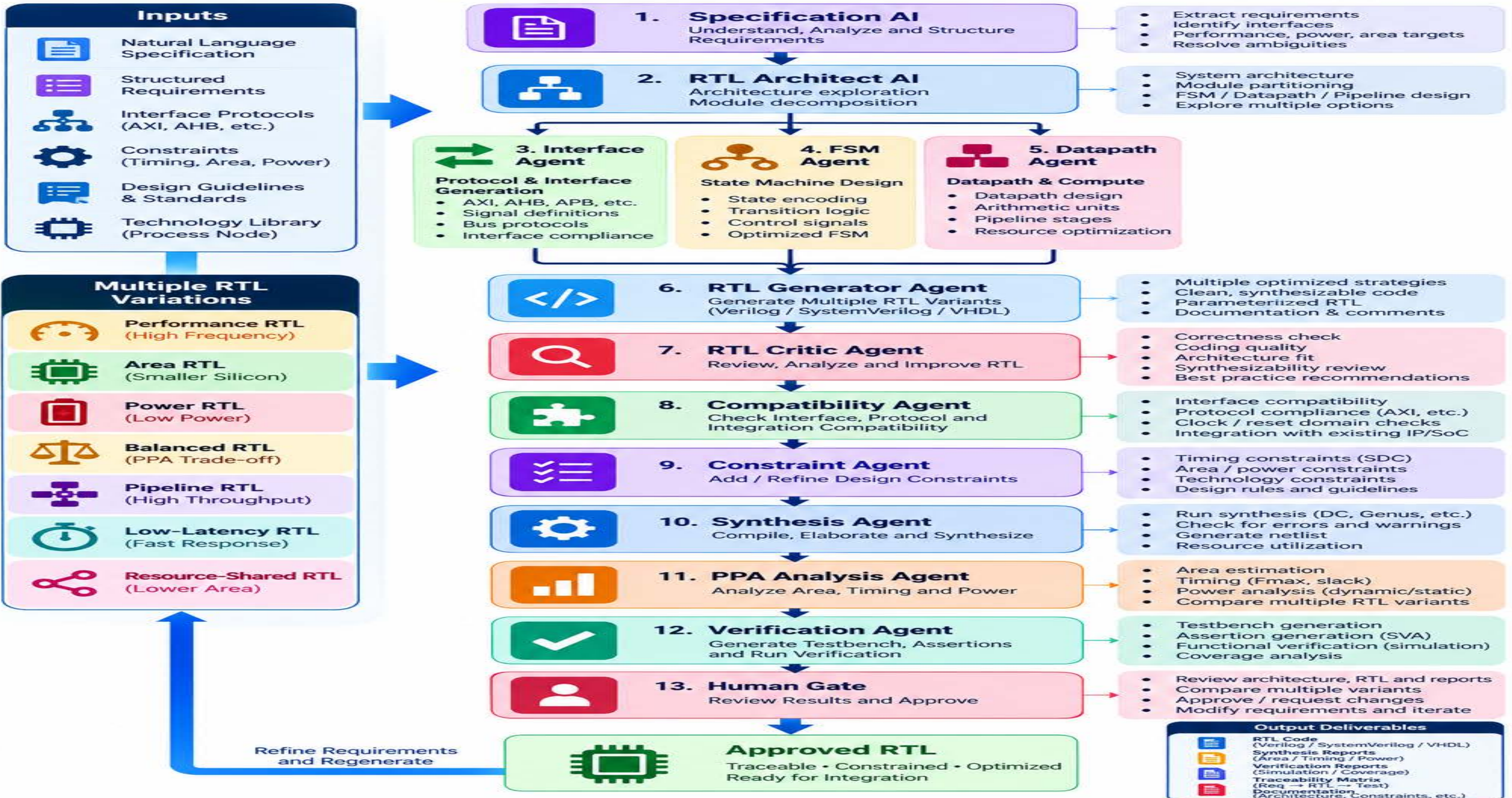
SiliconAI Spec2RTL — AI-Powered RTL Generation

**AI engineering system for traceable, constrained,
multi-objective RTL generation and refinement.**

Specification → Architecture → Generate → Critique → Map → Constrain → Compile → Synthesize → Analyze →
Human Approve → Refine → Regenerate



Specialized AI Agents Working Together for Traceable, Constrained and Optimized RTL Generation





A — Performance RTL

Max Frequency / Low Latency

Optimization Focus

- Frequency
- Latency
- Critical path reduction

Architecture Style (Example)

Deep Pipeline



- Deep pipelining
- Parallel execution
- Optimized critical path

Metrics (Typical)

Frequency	800 MHz (High)	High
Latency	Low	Low
Area	High	High
Power	High	High

Key Advantage

- ✓ Highest performance
- ✓ Maximum frequency

Trade-off

- ✗ More area
- ✗ Higher power

Typical Use Cases

- High-performance chips
- Compute intensive logic
- Real-time systems

B — Area RTL

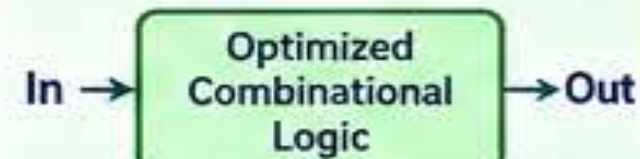
Minimize Gate / Resource Count

Optimization Focus

- Gate count
- Resource utilization
- Smaller silicon area

Architecture Style (Example)

Compact Logic



- Logic minimization
- Resource sharing
- Reduced redundancy

Metrics (Typical)

Frequency	Medium	Medium
Latency	Medium	Medium
Area	Low (Best)	Low
Power	Low	Low

Key Advantage

- ✓ Smaller silicon
- ✓ Lower gate count

Trade-off

- ✗ Potentially slower
- ✗ Lower throughput

Typical Use Cases

- Cost-sensitive designs
- Area-constrained SoCs
- Edge devices

C — Power RTL

Minimize Dynamic / Static Power

Optimization Focus

- Dynamic power
- Static power
- Power management

Architecture Style (Example)

Power Optimized



- Clock gating
- Power gating
- Low toggle design

Metrics (Typical)

Frequency	Medium	Medium
Latency	Medium	Medium
Area	Medium	Medium
Power	Low (Best)	Low

Key Advantage

- ✓ Low power
- ✓ Energy efficient

Trade-off

- ✗ Performance trade-offs
- ✗ May require design complexity

Typical Use Cases

- Battery-powered devices
- IoT / Edge AI
- Power-sensitive applications

D — Balanced RTL

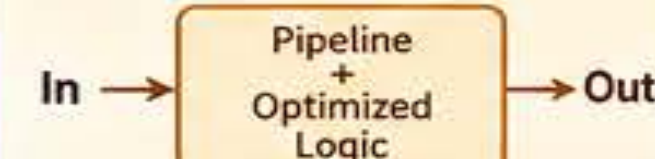
PPA Balance (Practical)

Optimization Focus

- Balanced PPA
- Practical implementation
- Good overall metrics

Architecture Style (Example)

Balanced Design



- Moderate pipelining
- Optimized logic
- Practical constraints

Metrics (Typical)

Frequency	Medium	Medium
Latency	Medium	Medium
Area	Medium	Medium
Power	Medium	Medium

Key Advantage

- ✓ Practical implementation
- ✓ Good PPA balance

Trade-off

- ✗ Not optimal in one dimension

Typical Use Cases

- General purpose SoCs
- Mainstream products
- Balanced requirements

E — Pipeline RTL

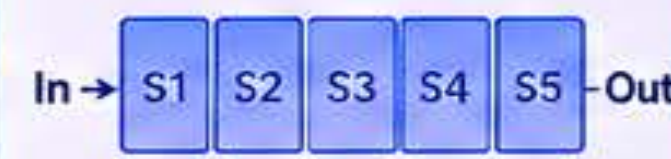
Max Throughput

Optimization Focus

- Throughput
- Parallel processing
- Pipeline depth

Architecture Style (Example)

Multi-Stage Pipeline



- Deep multi-stage pipeline
- High throughput design
- Parallel data processing

Metrics (Typical)

Frequency	High	High
Latency	Medium	Medium
Area	High	High
Power	High	High

Key Advantage

- ✓ High throughput
- ✓ Handles high data rates

Trade-off

- ✗ More registers/area
- ✗ Increased latency (pipeline)

Typical Use Cases

- High throughput systems
- Data processing / AI accelerators
- Streaming applications

F — Low-Latency RTL

Minimize Latency

Optimization Focus

- Latency
- Fast response
- Reduced stages

Architecture Style (Example)

Direct Path



- Shallow pipeline
- Optimized combinational path
- Reduced logic levels

Metrics (Typical)

Frequency	High	High
Latency	Low (Best)	Low
Area	High	High
Power	Medium	Medium

Key Advantage

- ✓ Fast response
- ✓ Low latency

Trade-off

- ✗ More hardware
- ✗ Higher area/power

Typical Use Cases

- Real-time control systems
- Communication systems
- Latency-critical applications

G — Resource-Shared RTL

Maximize Hardware Reuse

Optimization Focus

- Hardware reuse
- Time-multiplexing
- Resource sharing

Architecture Style (Example)

Shared Resources



- Time-multiplexed execution
- Shared functional units
- Control-driven scheduling

Metrics (Typical)

Frequency	Medium	Medium
Latency	Higher	Higher
Area	Low	Low
Power	Low	Low

Key Advantage

- ✓ Lower area
- ✓ Efficient resource utilization

Trade-off

- ✗ Lower parallelism
- ✗ Potentially slower

Typical Use Cases

- Area constrained designs
- Multi-function units
- Embedded / low-cost systems



Key Takeaways

- Multiple RTL implementations can be generated from the same specification.
- Choose the right variation based on your product goals (Performance, Area, Power, Throughput, Latency).
- AI can generate, evaluate and compare multiple variants automatically.
- Human-in-the-loop selects the best fit for the use case.

Comparison Summary

Variation	Optimization	Frequency	Latency	Area	Power	Throughput
A — Performance	Frequency / latency	High	Low	High	High	High
B — Area	Gate/resource count	Medium	Medium	Low	Low	Medium
C — Power	Dynamic/static power	Medium	Medium	Medium	Low	Medium
D — Balanced	PPA balance	Medium	Medium	Medium	Medium	Medium
E — Pipeline	Throughput	High	Medium	High	High	High
F — Low-Latency	Latency	High	Low	High	Medium	Medium
G — Resource-Shared	Hardware reuse	Medium	Higher	Low	Low	Lower



AI-Assisted RTL Exploration

- ✓ Generate multiple RTL variants automatically
- ✓ Evaluate PPA (Area, Timing, Power)
- ✓ Compare implementations side-by-side
- ✓ Explain trade-offs for each variant
- ✓ Select best variation based on requirements
- ✓ Refine with constraints and regenerate
- ✓ Human-in-the-loop approval

Variation	RTL Architecture (Illustration)	Optimization Focus	Key Advantage	Trade-off	Typical Use Cases
A – Performance RTL	Deep pipeline, parallel execution	- Frequency / latency - Critical path optimization	✓ Highest performance ✓ Maximum frequency	✗ More area ✗ Higher power	- High-performance SoCs - Compute intensive blocks - Real-time processing
B – Area RTL	Minimal logic, optimized structure	- Gate/resource count - Logic minimization	✓ Smaller silicon ✓ Lower gate count	✗ Potentially slower ✗ Lower throughput	- Cost-sensitive designs - Area-constrained SoCs - Edge devices
C – Power RTL	Power optimized design techniques	- Dynamic/static power - Clock/power gating	✓ Low power ✓ Energy efficient	✗ Performance trade-offs ✗ May require design complexity	- Battery-powered devices - IoT / edge AI - Power-sensitive applications
D – Balanced RTL	Balanced PPA approach	- PPA balance - Optimized for area, performance and power	✓ Practical implementation ✓ Good PPA balance	✗ Not optimal in one dimension	- General purpose SoCs - Mainstream products - Balanced requirements
E – Pipeline RTL	Multi-stage pipeline for high throughput	- Throughput - Parallelism	✓ High throughput ✓ Handles high data rates	✗ More registers/area ✗ Increased latency (pipeline)	- High throughput systems - Data processing accelerators - Streaming applications
F – Low-Latency RTL	Low latency design (minimal pipeline)	- Latency - Critical path reduction	✓ Fast response ✓ Low latency	✗ More hardware ✗ Higher area/power	- Real-time control systems - Communication systems - Latency-critical applications
G – Resource-Shared RTL	Time-multiplexed hardware reuse	- Hardware reuse - Time-multiplexing	✓ Lower area ✓ Efficient resource utilization	✗ Lower parallelism ✗ Potentially slower	- Area constrained designs - Multi-function units - Embedded / low-cost systems

1. SPECIFICATION

Natural Language Requirement

Functional Specification

SPEC-REQ-017

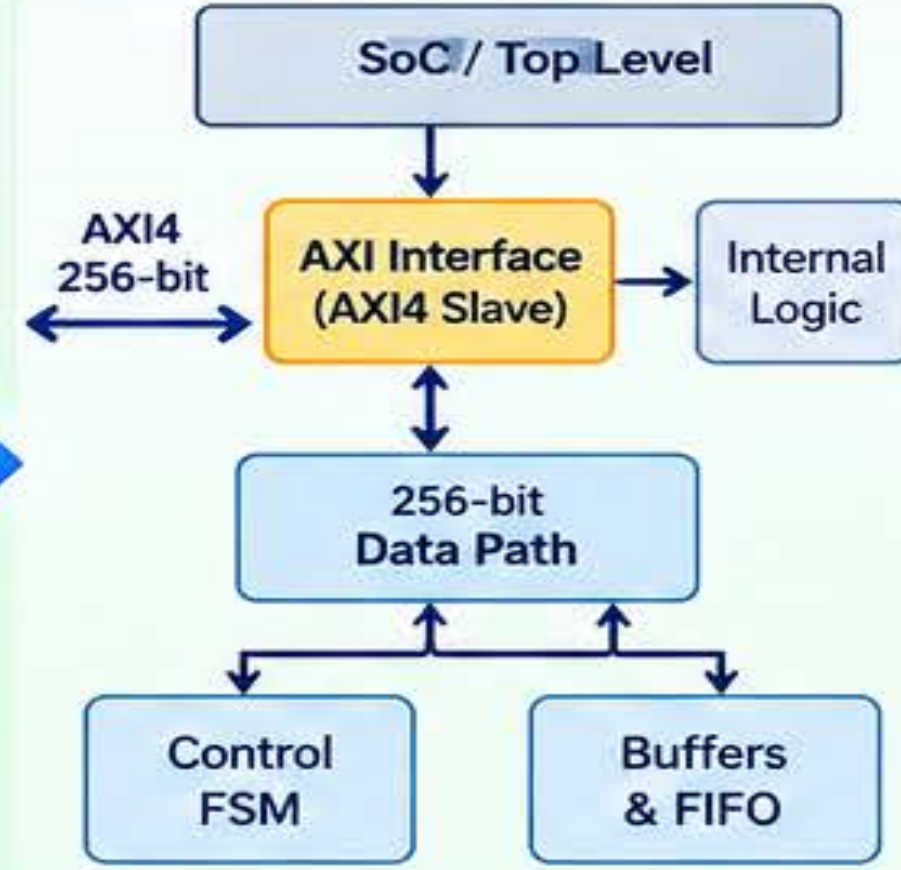
"Design shall support 256-bit AXI4 data."

Related Details

- Interface: AXI4
- Data Width: 256 bits
- Burst Length: 4K
- Outstanding Requests: 8
- ECC: Optional
- Low Power Mode: Required

2. ARCHITECTURE

Architectural Mapping



ARCHITECTURE REFERENCE

ARCH-AXI-02

- AXI4 Slave Interface
- 256-bit Data Path
- FSM for AXI Protocol
- Configurable Parameters

3. RTL IMPLEMENTATION

Generated Verilog / SystemVerilog

```
// AXI4 Interface (256-bit)
module axi_slave (
    input logic aclk,
    input logic aresetn,
    input logic [255:0] s_axi_wdata,
    input logic [31:0] s_axi_wstrb,
    input logic s_axi_wvalid,
    input logic s_axi_wready,
    output logic [255:0] s_axi_rdata,
    output logic s_axi_rvalid,
    input logic s_axi_rready
);
    ... // RTL implementation
endmodule
```

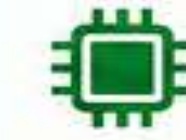
KEY SIGNALS

Signal Name	Width	Direction	Description
s_axi_wdata	[255:0]	Input	Write data bus
s_axi_wstrb	[31:0]	Input	Write strobe
s_axi_wvalid	1	Input	Write valid
s_axi_wready	1	Output	Write ready
s_axi_rdata	[255:0]	Output	Read data bus
s_axi_rvalid	1	Output	Read valid
s_axi_rready	1	Input	Read ready

4. SYNTHESIS

Implementation Results

256-bit Datapath Synthesis Results



Area
0.45 mm²
(120K gates)



Timing (Fmax)
500 MHz



Power
85 mW



Resource Utilization
- LUTs: 120K
- FFs: 95K
- RAM: 8 KB

Datapath View (Synthesis)



5. TRACEABILITY

End-to-End Mapping



SPEC-REQ-017
Design shall support 256-bit AXI4 data.



ARCH-AXI-02
AXI Interface 256-bit Data Path



MODULE-AXI-SLAVE
axi_slave



SIGNAL-s_axi_wdata
input [255:0]



RTL-LINE-142
s_axi_wdata declaration



TEST-AXI-017
Verify 256-bit data transfer

From RTL to Specification

Why does this RTL logic exist?

Bidirectional Explainability

From Specification to RTL

Show me exactly how this requirement is implemented?

Example Query 1: Specification → RTL



"Show me exactly how this specification requirement is implemented in RTL."

SPEC-REQ-017
Design shall support 256-bit AXI4 data.



ARCH-AXI-02
256-bit Data Path



RTL Code
s_axi_wdata [255:0]



Synthesis
256-bit datapath



Test
TEST-AXI-017

Example Query 2: RTL → Specification



"Why does this RTL logic exist?
Which specification requirement requires it?"

RTL Code
s_axi_wdata [255:0]



Module
AXI-SLAVE



Architecture
ARCH-AXI-02



Specification
SPEC-REQ-017
Design shall support 256-bit AXI4 data.

Human-in-the-Loop RTL Generation

AI-Assisted Engineering, Not Blind RTL Generation

- ✓ AI Generates Options
- ✓ Explains and Validates
- ✓ Engineer Makes the Right Decision
- ✓ Iterate to the Best RTL

Input

- Specification**
(Natural Language / Structured)
- Interfaces**
(AXI, AHB, etc.)
- Constraints**
(Timing, Area, Power)
- Design Guidelines**
(Coding Standards)
- Technology Library**
(Process Node)

Output

- Approved RTL**
(Verilog / SystemVerilog / VHDL)
- Synthesis Reports**
(Area / Timing / Power)
- Verification Assets**
(Testbench / Assertions)
- Traceability Matrix**
(Req → RTL → Test)
- Documentation**
(Architecture, Constraints, Analysis)



- AI Generates RTL**
Generate multiple RTL implementations from the specification

Variant A Performance Variant B Area Variant C Power ...



- AI Explains**
Explain architecture, design choices and specification mapping

AXI Interface → Data Path → Control FSM



- AI Critiques**
Analyze code quality, correctness, synthesizability and best practices

- ✓ Coding quality
- ✓ Synthesizability
- ✓ Architecture fit
- ⚠ Potential issues
- ⚠ Improvement suggestions



- AI Compares Alternatives**
Compare multiple RTL variants (Area, Timing, Power, Performance)

Variant	Area	Timing	Power	Latency
A	1.8 mm ²	800 MHz	120 mW	2 cycles
B	0.6 mm ²	400 MHz	80 mW	4 cycles
C	1.0 mm ²	500 MHz	40 mW	3 cycles
...	...	...	...	...



- AI Checks Compatibility**
Verify interface, protocol, clock, width and technology compatibility

- ✓ Interface (AXI4)
- ✓ Protocol Compliance
- ✓ Width/Clock Compatible
- ✓ Tech Node Supported



- AI Adds/Refines Constraints**
Add or refine timing, area, power and design constraints

```
create_clock -period 2.5 [get_ports clk]
set_input_delay 0.5 [get_ports axi_*]
set_max_area 1.0
set_max_power 0.0]
```



- AI Synthesizes**
Compile, elaborate and synthesize RTL with constraints

Synthesis → RTL → Synthesis Tool → Netlist Reports



- AI Reports PPA**
Provide detailed results with analysis and recommendations

Area: 0.75 mm² (-40%) Timing: 600 MHz (+50%) Power: 70 mW (-30%)



- Human-in-the-Loop**
Engineer reviews results and makes the decision

- Review architecture and RTL
- Check reports and comparisons
- Validate against product goals
- Modify requirements or constraints
- Select the best variant

Engineer Decision & Actions



- Approve**
Accept the current RTL implementation



- Reject**
Reject and request regeneration



- Modify Requirement**
Update functional or non-functional requirements



- Add Constraint**
Add or modify timing, area, power or technology constraints



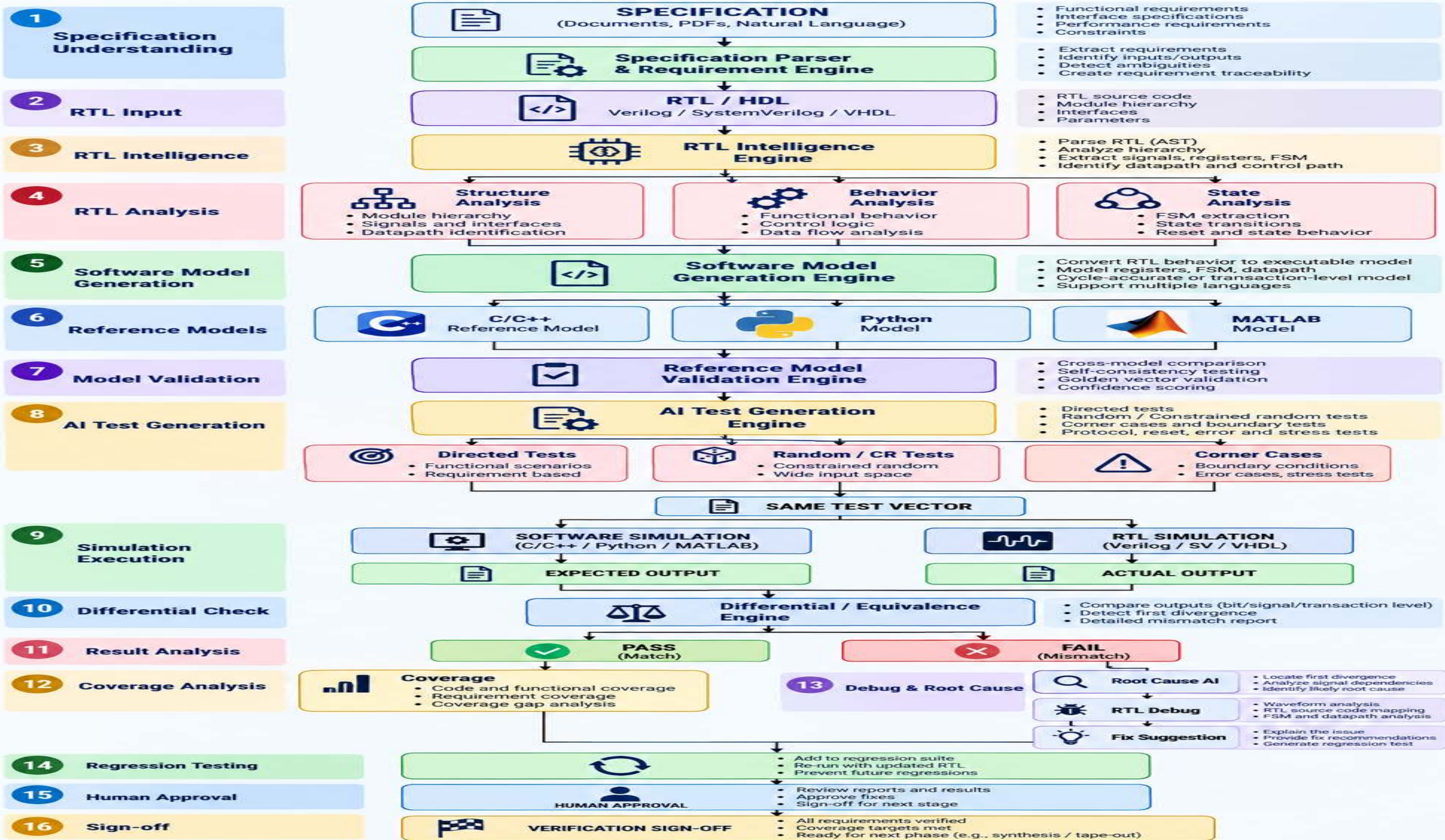
- Select Variation**
Choose the best RTL variant (performance / area / power / balanced)

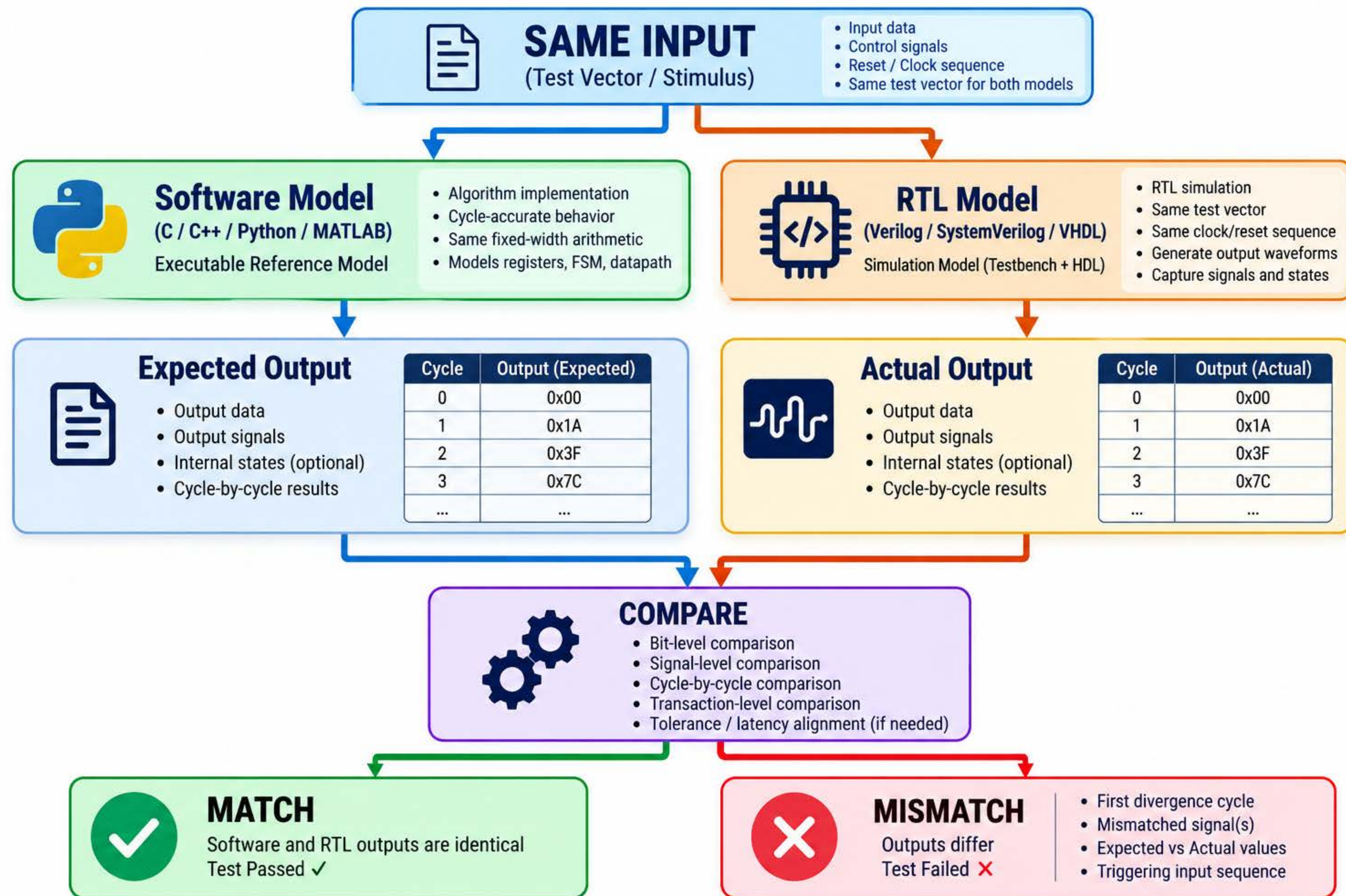


- Regenerate RTL**
AI incorporates human feedback and regenerates improved RTL

SiliconAI RTL2SIM

SiliconAI RTL2SIM is an AI-powered differential verification engine that creates executable software reference models from RTL/specifications, generates intelligent tests, compares software behavior against RTL behavior, and uses AI to identify and explain mismatches.





SiliconAI — AI-Powered Verification Intelligence

Understand the Specification. Understand the RTL. Generate the TestBench. Prove the Design. Find the Bugs. Close the Coverage Gaps.

SPEC → ARCHITECTURE → RTL → VERIFICATION PLAN → TEST GENERATION → TESTBENCH/UVM → SIMULATION/
FORMAL → COVERAGE → FAILURE → ROOT CAUSE → ADAPTIVE TEST GENERATION → VERIFICATION CLOSURE

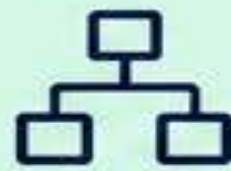
1 Specification



- Functional requirements
- Interface specifications
- Performance requirements
- Timing constraints
- Power requirements
- Use cases and scenarios

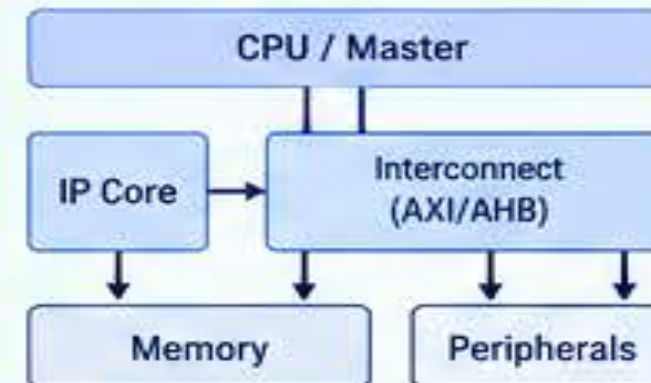
PDF / DOC
SRS
Datasheet
Algorithm
Requirements

2 Architecture



- Block diagram
- Module hierarchy
- Not! Interfaces (e.g. AXI, AHB, Custom)
- Data flow
- Control flow
- FSM / State diagrams

SoC / IP Architecture



3 RTL Code



- Verilog / SystemVerilog / VHDL
- Module hierarchy
- Registers, FSM
- Datapath and control logic
- Parameters and configuration
- Interface definitions

```
module fifo (  
    input  logic clk,  
    input  logic rst_n,  
    input  logic wr_en,  
    input  logic rd_en,  
    input  logic [7:0] din,  
    output logic [7:0] dout,  
    output logic full,  
    output logic empty  
);  
...  
endmodule
```

4 Reference Model



C / C++ / Python / MATLAB
Golden / Behavioral Model

5 Coverage Information



- Existing coverage
- Uncovered features
- Uncovered FSM states
- Uncovered scenarios
- Code / functional coverage

6 Previous Failures



- Bug reports
- Failure waveforms
- Root cause analysis
- Corner cases missed
- Regression history



AI Test Generation Engine Intelligent Verification Brain

Understand Requirements

Analyze RTL & Architecture

Identify Functional Scenarios

Generate Test Strategy

Create Stimulus & Expected Results

Target Coverage Gaps

Learn from Previous Failures



LLM + EDA Knowledge



Requirement-to-Test Mapping



Coverage-Driven Test Generation



Constraint Solving (Random/CR)



Protocol & Interface Awareness



FSM & State Exploration



Corner Case Reasoning



Power/Performance Scenarios



Self-Validation of Tests



Generated Test Categories



Directed Tests

Functional scenarios
Requirement based



Random Tests

Wide input space



Constrained Random

With protocol/field constraints



Boundary Tests

Min/max values, limits



Corner Cases

Rare and critical cases



Negative Tests

Invalid inputs, error conditions



Stress Tests

Long run, high load scenarios



Protocol Tests

AXI/AHB/Custom protocol checks



FSM Tests

All states and transitions



Reset Tests

Reset behavior, sequence



Error Tests

Error handling, exceptions



Overflow Tests

Arithmetic overflow/underflow



Latency Tests

Timing and latency checks



Performance Tests

Throughput, bandwidth, QoS



Power Tests

Low power modes, power states



Regression Tests

From previous failures & fixes



Generated Test Bench

Complete Verification Environment

- Self-checking testbench
- Stimulus generators
- Protocol drivers (AXI/AHB/UVM)
- Scoreboard & checkers
- Assertions (SVA)
- Functional coverage
- Configurable and reusable

```
// Testbench template  
module tb_top;  
    // Clock/Reset  
    // DUT instantiation  
    // Stimulus generation  
    // Scoreboard  
    // Assertions  
    // Functional coverage  
    // ...  
endmodule
```



Generated Test Cases

Detailed and Traceable Test Set

Test ID	Input Stimulus	Expected Behavior	Coverage Target	Req ID	Priority
TC_001	Reset sequence	All registers	100%	REQ-001	High
TC_002	Write/Read basic	Data match	95%	REQ-010	High
TC_003	FIFO full condition	full=1	100%	REQ-012	High
TC_004	FIFO empty condition	empty=1	100%	REQ-013	High
TC_005	Boundary values	Min/Max	90%	REQ-020	Medium
TC_006	Invalid input	Error response	85%	REQ-021	Medium
TC_007	Burst transfer	Data integrity	95%	REQ-030	High
TC_008	Latency check	Within N cycles	90%	REQ-040	Medium
TC_009	Power mode	Low power state	80%	REQ-050	Low
...	...	...	...	...	...



Verification Coverage Targets



Functional Coverage
(features, states, transitions)



Code Coverage
(statement, branch, toggle)



Requirement Coverage
(Req → Test mapping)



Interface / Interoperability
(AXI/AHB/Custom)



Timing & Latency
(setup, hold, latency, throughput)



Power & Low Power
(power states, transitions)



Error & Exception Handling



Regression Coverage



Outputs & Reports



Testbench (UVM / SV / VHDL)



Test cases with expected results



Requirement traceability matrix



Coverage reports



Simulation scripts



Waveform analysis



Pass/Fail summary



Root cause analysis (for failures)



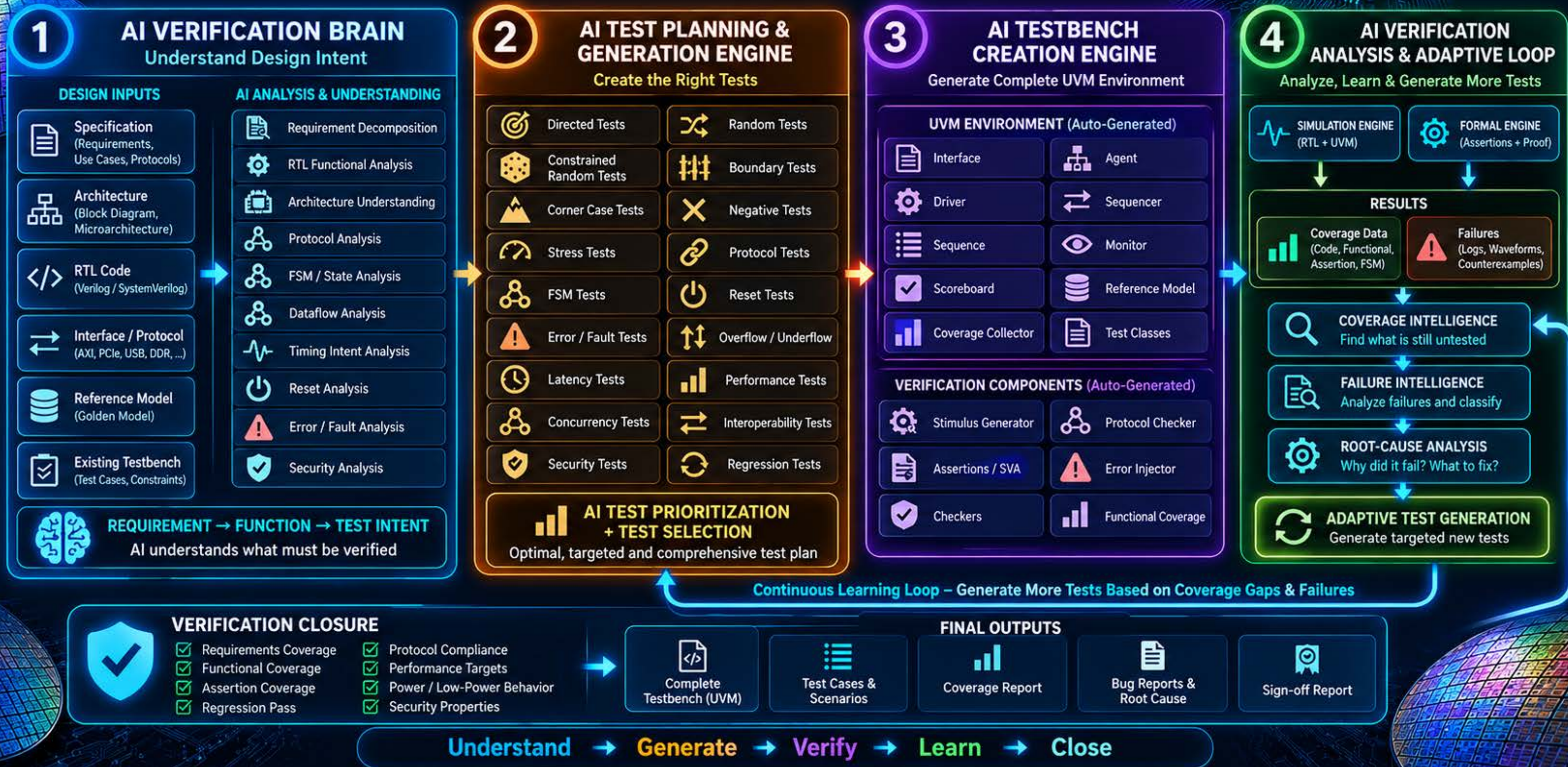
Regression test suite

SILICON AI — AI TEST GENERATION & VERIFICATION

4 Core Intelligence Engines

From Specification to Verification Closure — AI-Powered, End-to-End, Autonomous

FASTER
COVERAGE
FEWER BUGS
HIGHER QUALITY
SILICON SUCCESS



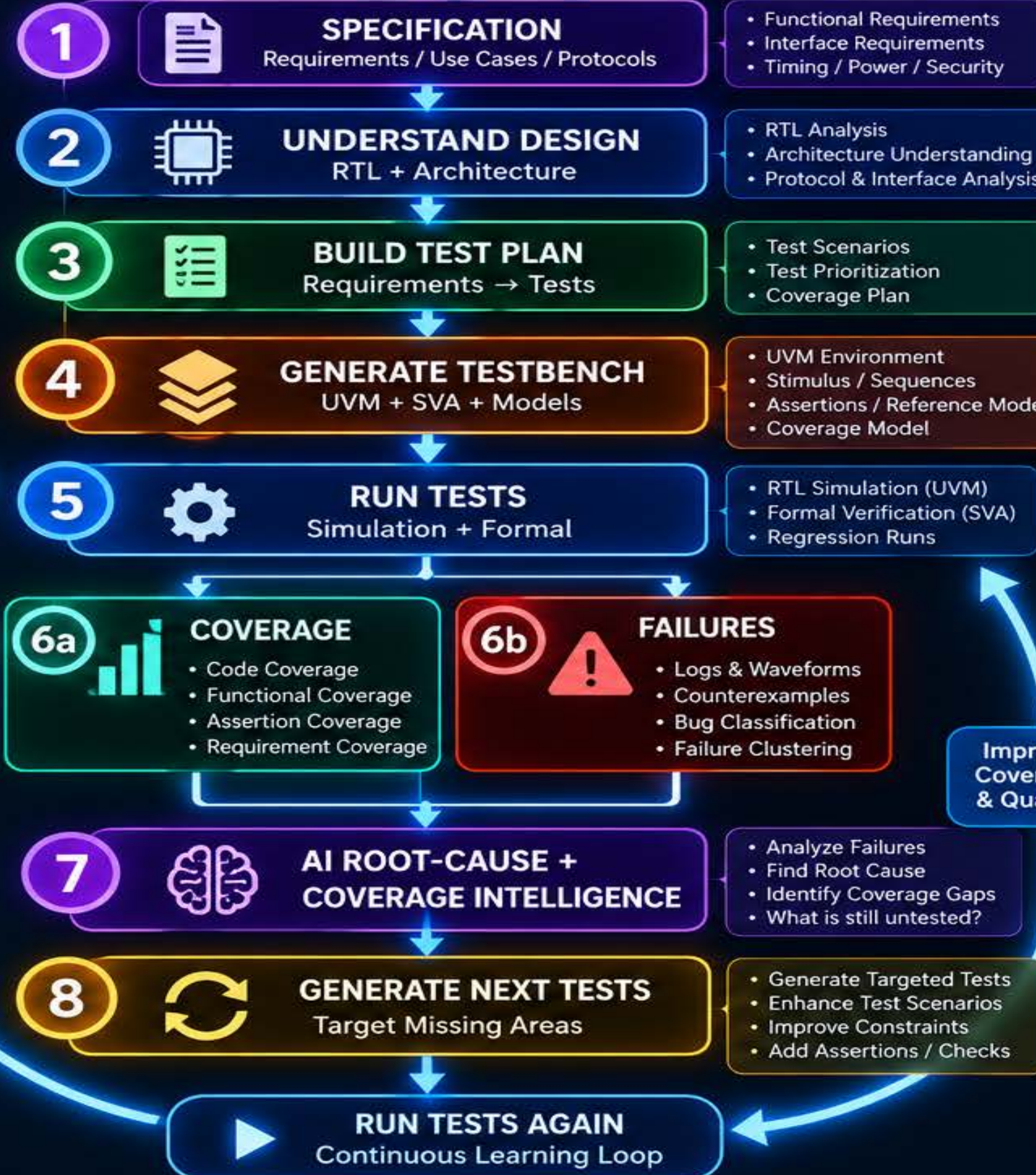
AI Test Generation Loop

Self-Improving Verification from Specification to Closure

AI doesn't just generate tests.
It understands what must be verified, generates the right tests, executes them, analyzes failures and coverage gaps, and continuously creates new tests until verification goals are achieved.

INPUTS

- Specification**
Requirements, Use Cases, Protocols
- Architecture**
Block Diagram, Interfaces, Microarchitecture
- RTL Code**
Verilog / SystemVerilog
- Interface / Protocol**
AXI, PCIe, USB, DDR ...
- Reference Model**
Golden Model
- Existing Testbench**
Test Cases, Constraints
- Coverage Data**
Functional, Code, Assertion
- Simulation Results**
Logs, Waveforms
- Previous Failures**
Bug History



OUTPUTS

- Complete UVM Testbench**
Environments, Sequences, Assertions
- Test Cases & Scenarios**
Directed, Random, Constrained, ...
- Coverage Reports**
Code, Functional, Assertion, FSM
- Simulation Results**
Logs, Waveforms, Metrics
- Bug Reports & Root Cause**
Failure Analysis & Fix Recommendations
- Sign-off Report**
Requirements, Coverage, Quality

VERIFICATION CLOSURE

- ✓ Requirements Coverage
- ✓ Functional Coverage
- ✓ Assertion Coverage
- ✓ Regression Pass
- ✓ Protocol Compliance
- ✓ Performance Targets
- ✓ Power / Low-Power Behavior
- ✓ Security Properties
- ✓ Interoperability
- ✓ All Critical Tests Passed

Self-Improving Verification

Continuously learn from results and generate better tests until verification goals are achieved.

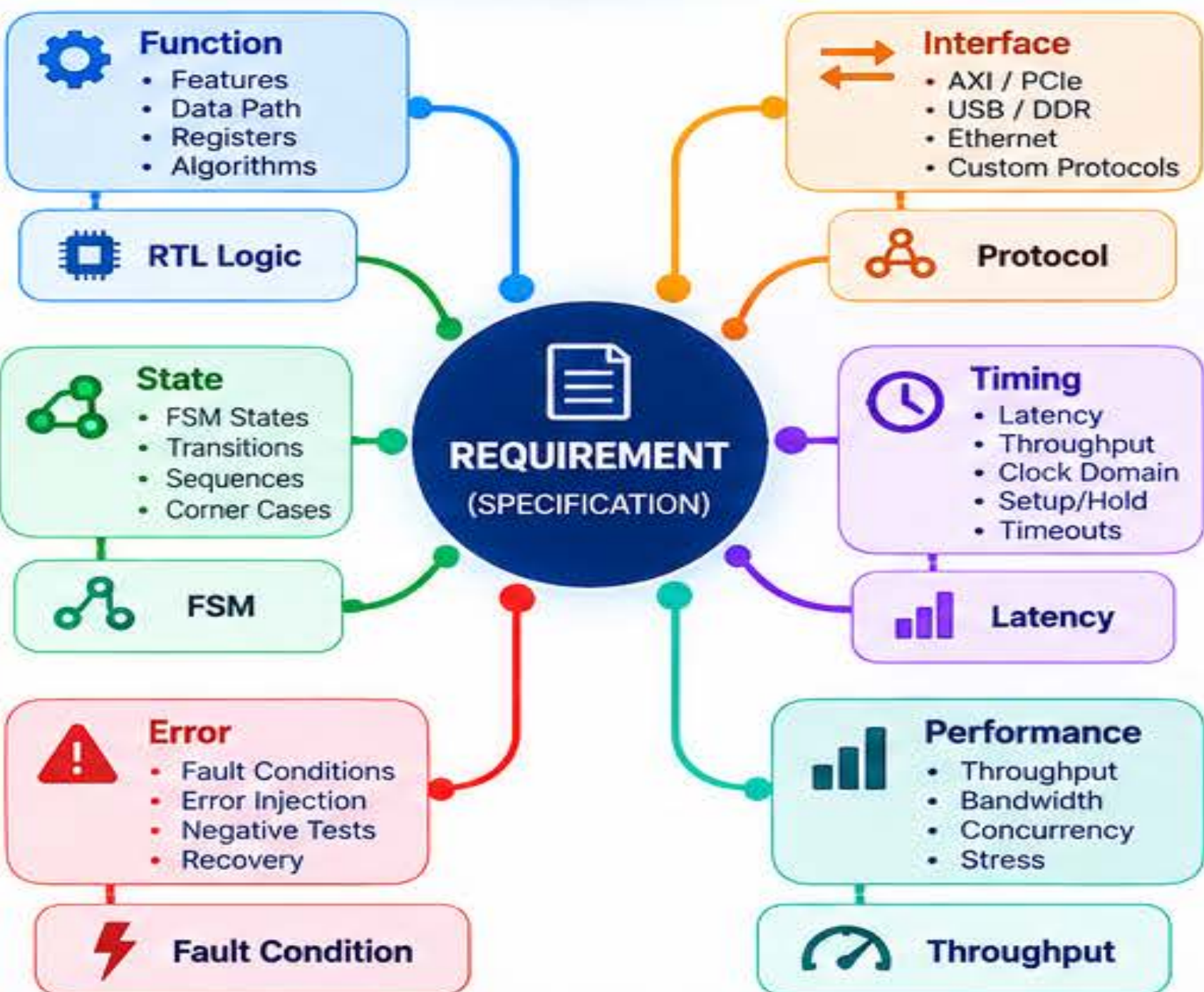
From Specification
to Verification Closure
with AI Intelligence

Test Generation Should Be Requirement-Driven

AI creates a Verification Knowledge Graph, maps requirements to tests, and ensures full traceability.

1 VERIFICATION KNOWLEDGE GRAPH

AI understands the design and builds a semantic model



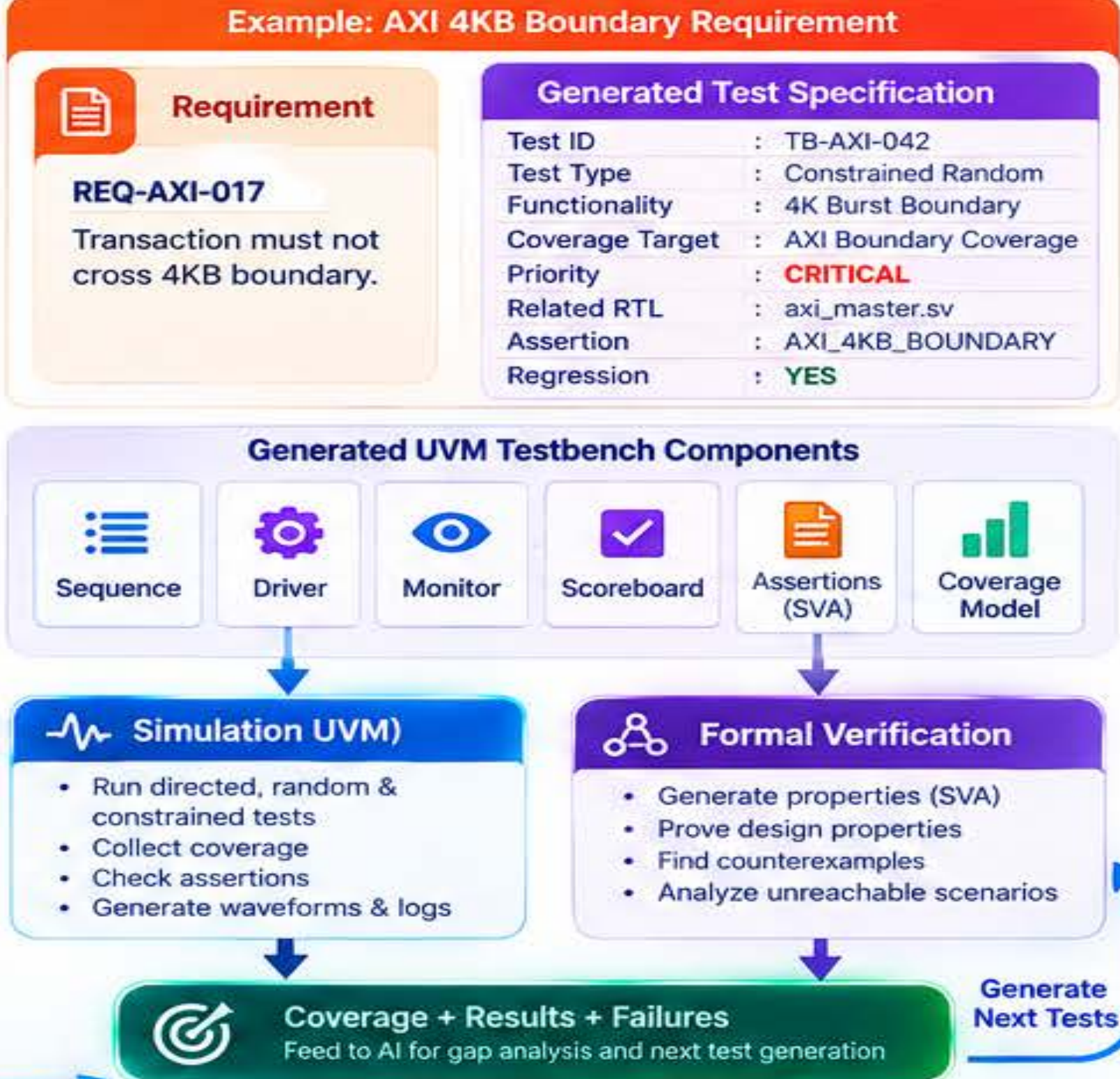
2 REQUIREMENT → TEST FLOW

AI creates a complete verification chain



3 AI-GENERATED TEST FROM REQUIREMENT

End-to-end generation with full traceability



4 FULL TRACEABILITY FROM REQUIREMENT TO TEST IMPLEMENTATION

Every test is linked to requirement, coverage, assertion and testbench components

Requirement ID	Feature / Functionality	Verification Objective	Coverage Point	Test Scenario	Stimulus (Example)	Expected Result	Assertion / Checker	Testbench Component	Test ID	Priority	Status
REQ-AXI-017	4KB Boundary	No transaction across 4KB	AXI_BOUNDARY_COV	Constrained Random	Addr=0x0FFC Len=16, Size=256b	Must not cross 4KB	AXI_4KB_BOUNDARY	Seq, Driver, Monitor, Scoreboard, Coverage	TB-AXI-042	CRITICAL	Generated
REQ-AXI-031	Write Read Ordering	Maintain ordering	AXI_ORDER_COV	Directed	Back-to-back writes/reads	Data returned in order	AXI_ORDER_CHECK	Sequence, Scoreboard	TB-AXI-055	HIGH	Generated
REQ-FIFO-009	No read when empty	Prevent underflow	FIFO_UNDERFLOW_COV	Negative Test	Read when empty	rd_valid must be 0	FIFO_NO_READ_EMPTY	Sequence, Assertions	TB-FIFO-021	CRITICAL	Generated
REQ-CLK-002	Reset Behavior	Correct reset operation	RESET_COV	Reset Test	Assert reset during Xact	Design in reset state	RESET_ASSERTION	Sequence, Monitor	TB-CLK-008	HIGH	Generated
REQ-PWR-005	Low Power Mode	Enter/exit low power	POWER_STATE_COV	Power Test	Sleep → Active	Correct state transition	POWER_STATE_CHECK	Sequence, Checker	TB-PWR-012	MEDIUM	Generated

Every Test Gets an AI-Generated Test Contract

From Requirement → Test Contract → Complete Testbench (UVM + SVA + Coverage + Expected Results)

1 INPUTS TO AI

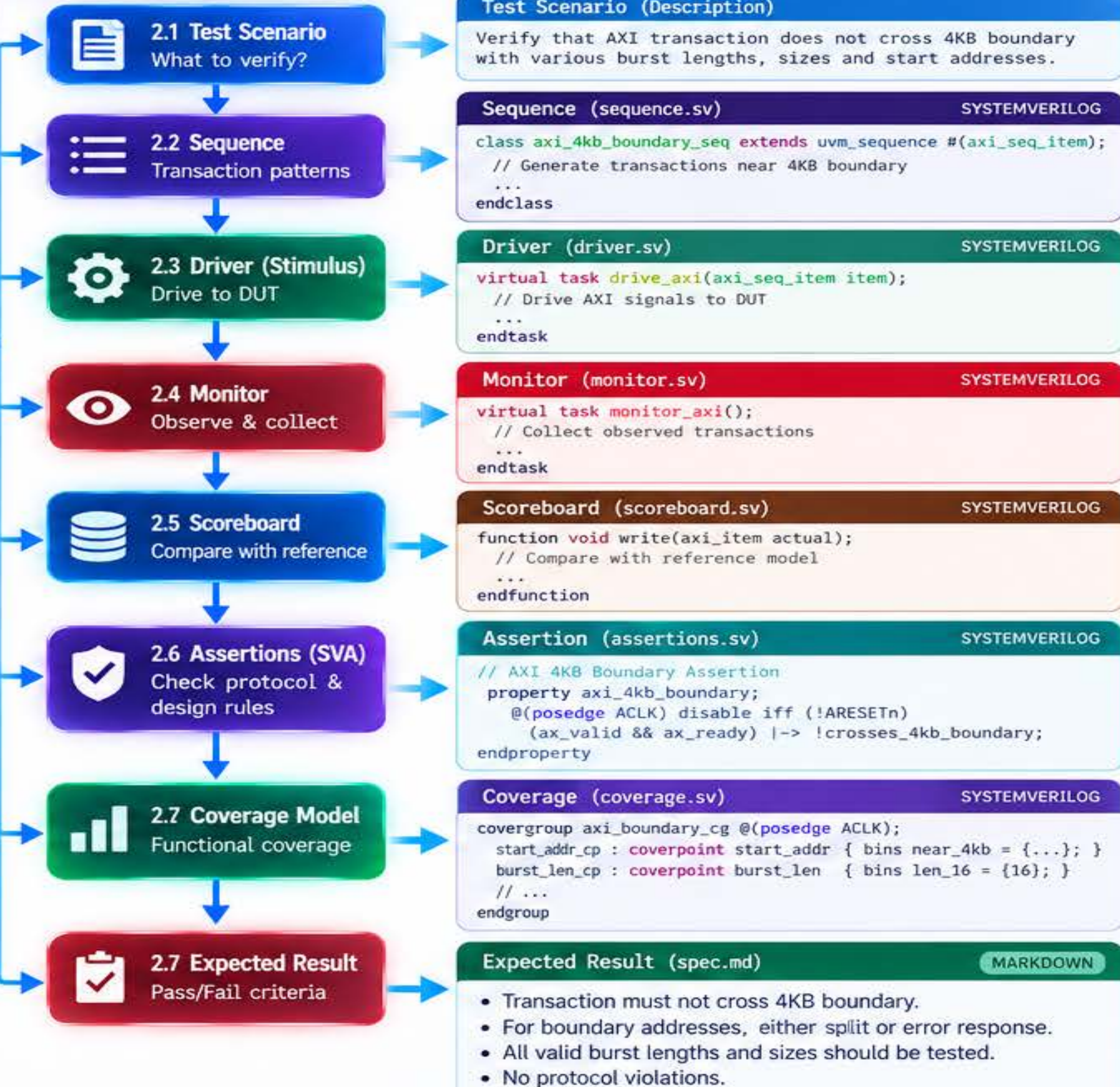
- Specification**
Requirements, Use Cases
- Architecture**
Block Diagram, Interfaces
- RTL Code**
Verilog / SystemVerilog
- Interface / Protocol**
AXI, PCIe, USB, DDR ...
- Reference Model**
Golden Model / Algorithmic
- Existing Testbench**
Tests, Constraints, Sequences
- Coverage Data**
Coverage Reports
- Previous Failures**
Bug History & Fixes

AI
ANALYZES
&
GENERATES

AI-GENERATED TEST CONTRACT

TEST CONTRACT ✓	
Test ID	TB-AXI-042
Requirement	REQ-AXI-017
Feature	4KB Boundary
Test Type	Boundary + Negative
Priority	CRITICAL
RTL Module	axi_master.sv
Interface	AXI4
Stimulus	Generated
Expected Result	Generated
Assertion	AXI_4KB_BOUNDARY
Coverage	AXI_BOUNDARY_COV
Regression	YES
Status	GENERATED

2 GENERATED TEST COMPONENTS (UVM + SVA + COVERAGE)



3 EXAMPLE: AXI 4KB BOUNDARY TEST

Test Scenario

- Start address near 4KB boundary (e.g., 0x0FFC)
- Different burst lengths (1, 4, 8, 16)
- Different data sizes (32/64/128/256-bit)
- INCR, WRAP, FIXED bursts
- Verify no transaction crosses 4KB boundary

Stimulus Example

Address = 0x0FFC
Burst = INCR
Length = 16
Size = 256-bit

Waveform (Simplified)

ACLK	[Clock Signal]
ARESETn	[Reset Signal]
AWADDR	0x0FFC
AWLEN	16
AWSIZE	256-bit
AWBURST	INCR
AWVALID	[Valid Signal]
AWREADY	[Ready Signal]

Coverage Results

AXI Boundary Coverage: **92%**

- Start Address Coverage
- Burst Length Coverage
- Data Size Coverage
- Boundary Conditions
- Protocol Checks

Simulation Results

PASS
No 4KB boundary violation
All assertions passed

Key Assertions

- AXI_4KB_BOUNDARY
- AXI_PROTOCOL_CHECK
- AXI_VALID_READY
- AXI_RESET_BEHAVIOR

Verification Dimensions

Complete, End-to-End Verification for High-Quality Silicon

Broader than Functional Testing
AI verifies functionality, protocols, timing, power, security and more to ensure high-quality, reliable silicon.

1 FUNCTIONAL VERIFICATION

Does it work as per the specification?

- Features & Use Cases
- Data Path Verification
- Registers & CSRs
- Algorithms & Processing
- Control Logic
- End-to-End Scenarios

Example Tests

- Read/Write Operations
- Data Processing
- Register Configuration
- Algorithm Correctness
- Mode Transitions
- System Level Tests

2 PROTOCOL VERIFICATION

Does it follow interface and protocol standards?

- AXI / AXI4 / AXI5
- PCIe
- USB
- DDR / LPDDR
- Ethernet
- Custom Interfaces

Example Tests

- Protocol Compliance
- Transaction Ordering
- Boundary Conditions
- Error Scenarios
- Interoperability
- Multi-Protocol Systems

3 STRUCTURAL VERIFICATION

Is the design structurally correct and robust?

- FSM / State Machines
- CDC / RDC (Clock/Reset)
- Reset Verification
- Connectivity Check
- Design Rule Checks
- Lint & Formal Structural Checks

Example Tests

- State Transitions
- Reset During Operation
- Clock Domain Crossings
- Deadlock Analysis
- Unreachable States
- Connectivity & Integrity

4 PERFORMANCE VERIFICATION

Does it meet performance and timing goals?

- Latency
- Throughput
- Concurrency
- Stress Testing
- Timing Verification
- Scalability

Example Tests

- High Throughput Scenarios
- Back-to-Back Transactions
- Multi-Thread / Multi-Master
- Stress & Soak Tests
- Timeout / Latency Checks
- Performance Under Load

5 QUALITY VERIFICATION

Is the design verified with sufficient quality?

- Code Coverage
- Functional Coverage
- Assertion Coverage
- Regression Testing
- Maintainability
- Traceability (Req → Test)

Example Tests

- Coverage Closure
- Missing Scenario Analysis
- SVA Property Checks
- Regression Suites
- Reusability & Modularity
- Requirement Traceability

6 RESILIENCE VERIFICATION

How does it behave under abnormal conditions?

- Error / Fault Handling
- Fault Injection
- Negative Testing
- Corner Cases
- Robustness
- Recovery & Self-Healing

Example Tests

- Invalid Transactions
- Error Responses
- Power/Clock Glitches
- Boundary Conditions
- Corner Case Stimulus
- Recovery & Reinitialization

7 POWER / SECURITY VERIFICATION

Is the design power efficient and secure?

Power Verification

- Power Modes (Active/Idle/Sleep)
- Clock Gating / Power Gating
- DVFS (Dynamic Voltage Frequency Scaling)
- Power State Transitions
- Retention & Isolation
- Low Power Protocols

Example Tests

- Active → Idle → Sleep
- Wake-up Sequence
- Power Domain Isolation
- Retention Register Check
- Low Power Compliance

Security Verification

- Security Threat Analysis
- Access Control & Permissions
- Secure Boot / Secure Modes
- Data Integrity / Encryption
- Side-Channel Considerations
- Security Protocols

Example Tests

- Invalid Access Attempts
- Security Violation Checks
- Encrypted Data Handling
- Privilege Escalation Tests
- Protocol Security Tests

Low Power Verification

- Low Power Behavior
- Sleep / Deep Sleep / Wake
- Power Domain Management
- Isolation & Retention
- Power Sequencing
- Power-Performance Tradeoff

Example Tests

- Sleep Mode Entry/Exit
- Wake-up Latency
- Isolation Cell Verification
- Retention Data Check
- Multiple Power States



Add Timing, Power and PPA Carefully

AI generates targeted verification scenarios for Timing and Power, and provides PPA analysis and optimization as a feedback loop.

1 TIMING VERIFICATION

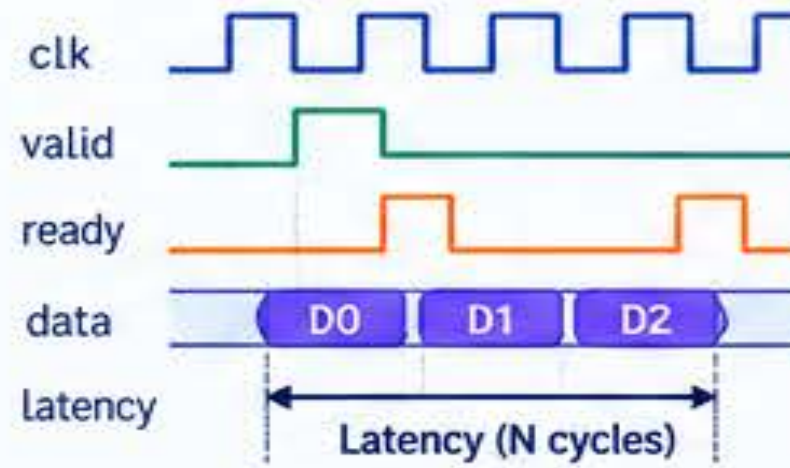
AI generates timing-focused test scenarios



Key Test Scenarios

- Clock relationships**
Multiple clock domains, frequency ratios
- Latency**
End-to-end latency, pipeline latency
- Throughput**
Sustained bandwidth, peak performance
- Setup/Hold scenarios**
Timing corner cases, margin checks
- Backpressure**
Flow control, stalls, congestion
- Timeout**
Response time, watchdog timers
- Clock-domain interactions**
CDC behavior, synchronization
- Protocol timing**
AXI/PCIe/DDR timing compliance

Example: Latency Test Scenario



Generated Checks

- ✓ Latency within N cycles
- ✓ Setup/Hold timing checks
- ✓ No protocol timing violations
- ✓ Backpressure handling
- ✓ Timeout conditions
- ✓ Multi-clock domain behavior

2 POWER VERIFICATION

AI generates power and low-power test scenarios



Key Test Scenarios

- Power-state transitions**
All power mode transitions
- Idle → Active**
Wake-up latency, state restore
- Active → Sleep**
Save state, power down
- Wake-up**
Correct wake-up sequence
- Clock gating**
Gating enable/disable checks
- Power gating**
Domain isolation, power control
- Retention**
State retention across power cycles
- Isolation**
Isolation cell behavior
- Low-power protocols**
UPF, power intent, low-power modes

Example: Power State Transition

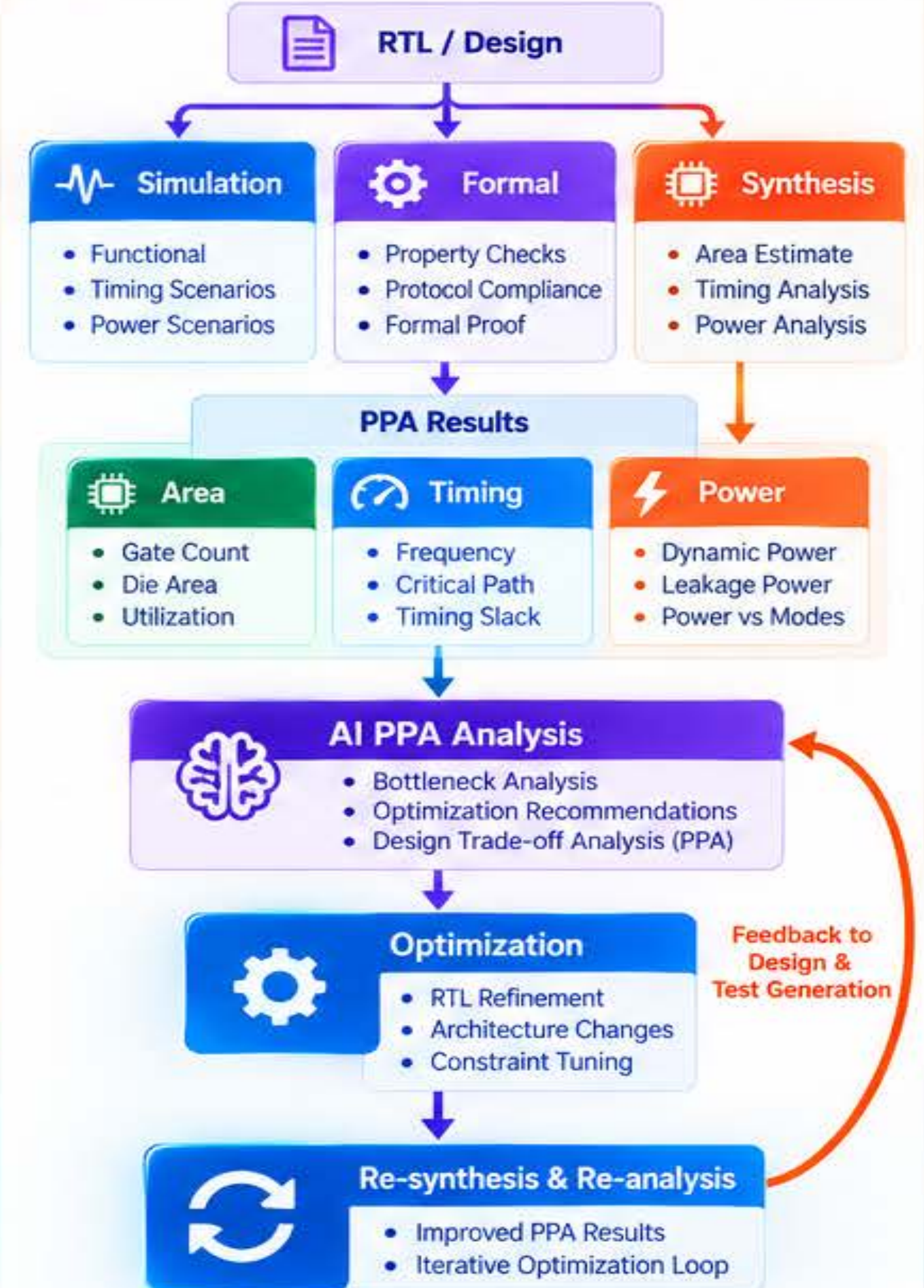


Generated Checks

- ✓ Correct power state transitions
- ✓ Clock gating effectiveness
- ✓ Isolation during power down
- ✓ State retention after wake-up
- ✓ Wake-up latency within limit
- ✓ UPF/Low-power intent compliance

3 PPA ANALYSIS & OPTIMIZATION

PPA is an analysis and sign-off feedback domain



4 INTEGRATED FLOWS: TEST GENERATION, VERIFICATION AND PPA FEEDBACK

Timing and Power scenarios are generated by the AI Test Generation Engine. PPA analysis provides feedback for optimization.



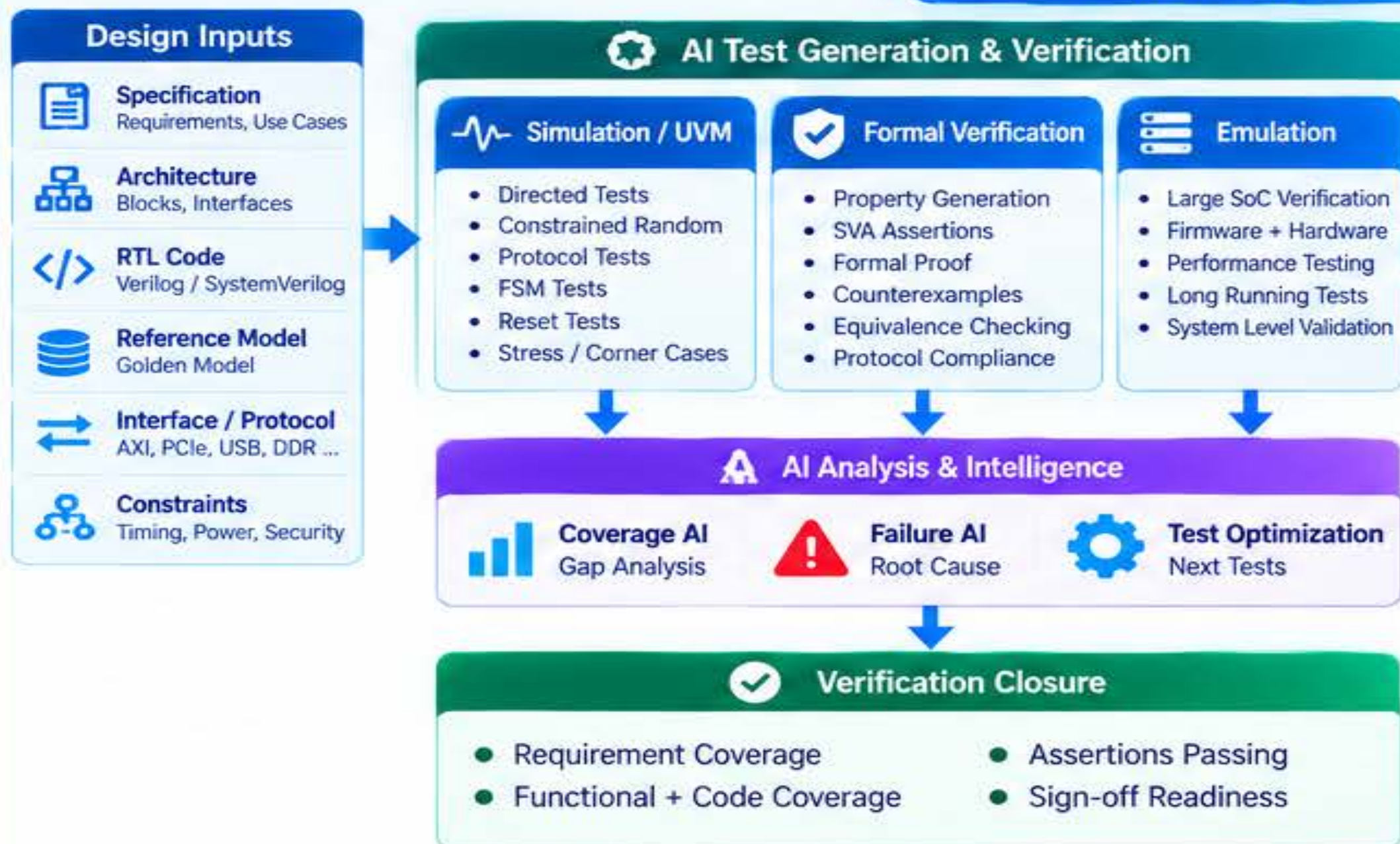
Iterate Until Verification and PPA Goals Are Achieved

1 PRE-SILICON VERIFICATION

Prove the design before fabrication



- ✓ Functional Correctness
- ✓ Protocol Compliance
- ✓ Performance, Power, Security
- ✓ Coverage Closure

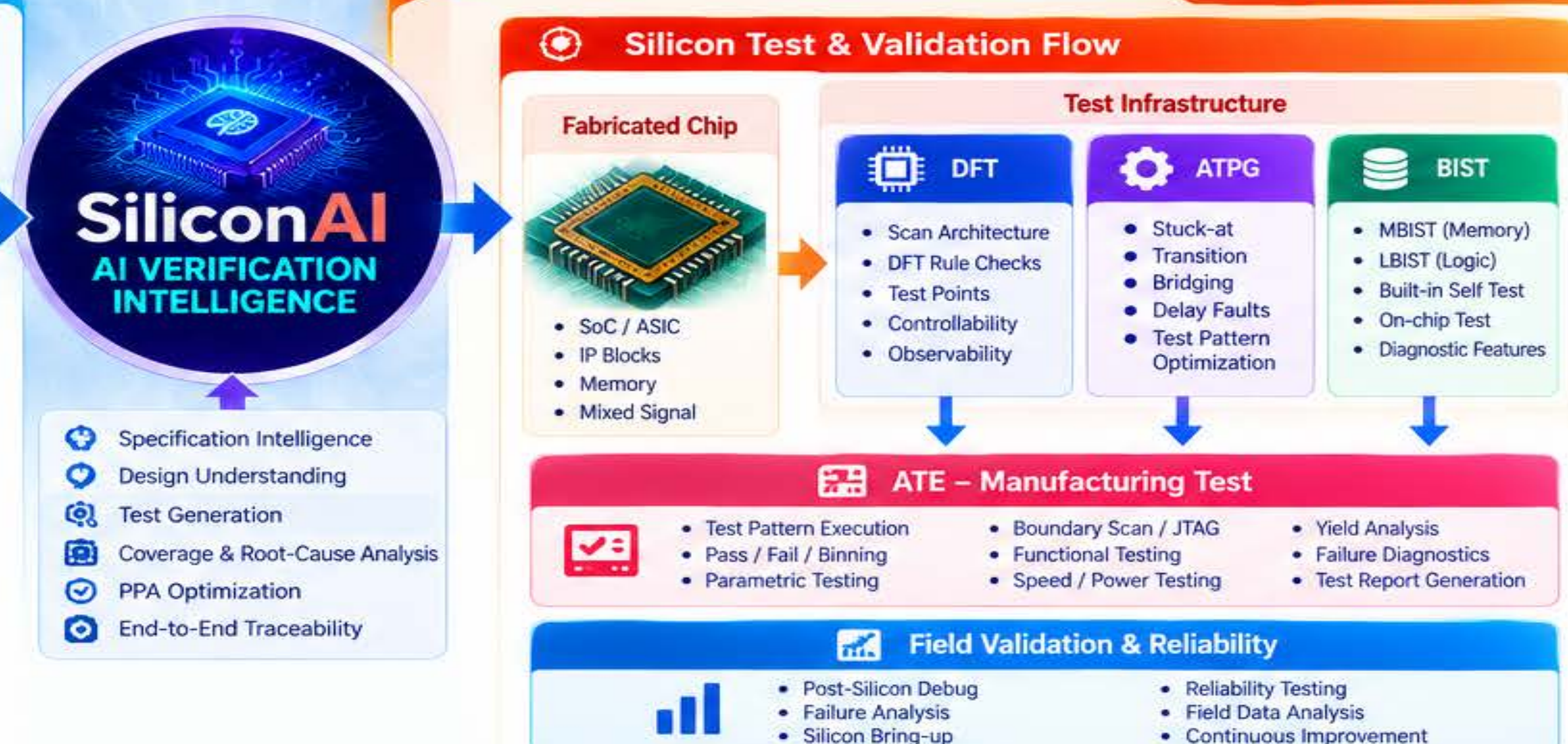


2 POST-SILICON TEST

Validate the manufactured silicon



- ✓ Manufacturing Test
- ✓ Yield Improvement
- ✓ Field Reliability
- ✓ Failure Analysis



3 TRADITIONAL VERIFICATION PROCESS

Manual, time-consuming and iterative



4 SILICONAI – AI-DRIVEN VERIFICATION PROCESS

End-to-end intelligent and self-improving



5 KEY BENEFITS

- Reduced Verification Time (up to 50-80%)
- Higher Coverage & Quality
- AI-Generated Tests & Testbench
- PPA Optimization Feedback Loop
- Better Silicon Reliability
- End-to-End Traceability
- Scalable from IP to Full SoC

SiliconAI RTL Intelligence

AI Understands, Debugs, Explains, Optimizes & Evolves Existing RTL

SiliconAI RTL2SIM is an AI-powered differential verification engine that creates executable software reference models from RTL/specifications, generates intelligent tests, compares software behavior against RTL behavior, and uses AI to identify and explain mismatches.



AI analyzes existing RTL to deliver debug, insights and higher quality RTL.



DEBUG RTL - FLOW

From Failure to Fix with AI



DEBUG RTL - KEY CAPABILITIES

- Code Analysis
- Code Coverage
- Critical Issues Detection
- Root-Cause Analysis
- Waveform / Simulation Analysis
- Assertion & Verification Analysis
- Bug Detection & Fix Suggestions



RTL INTELLIGENCE - CORE CAPABILITIES

- Code Review
- Lint Explanation
- Refactoring
- Optimization
- Legacy RTL Understanding
- RTL → Specification
- Interface Analysis
- Dependency Analysis

1 RTL CODE (Buggy Example)

```

counter.v
1 module counter (
2   input clk,
3   input rst_n,
4   output reg [7:0] count
5 );
6 always @(posedge clk) begin
7   if (!rst_n)
8     count <= 8'd0;
9   else if (en)
10    count <= count + 1;
11 end
12 endmodule
    
```

Simulation Error Log

```

# Error: counter_tb.v(45):
# Assertion failed: count should
# not increment when en=0
# Time: 250 ns
# Fatal: Simulation stopped.
    
```

2 WAVEFORM (VCD/FSDB)



ASSERTION (SVA)

```

1 property p_no_increment;
2   @(posedge clk)
3   disable iff (!rst_n)
4   !en |-> $stable(count);
5 endproperty
    
```

Test Case Result

- Assertion failed
- count changed when en = 0
- Time: 250 ns

3 AI ANALYSIS & ROOT CAUSE

- Code & Lint Analysis**
Detects coding errors, lint issues, unsafe constructs and bad practices.
- Waveform Analysis**
Finds first divergence and correlates with source code.
- Coverage Analysis**
Identifies untested scenarios and low coverage areas.
- Root Cause Identification**
Traces failure to the exact RTL lines and condition responsible.
- Related Issues (Knowledge Graph)**
Finds similar issues and patterns from past designs and public data.

4 AI SUGGESTED FIX & REFINEMENT

```

Fixed RTL Code      counter_fixed.v
1 module counter (
2   input clk,
3   input rst_n,
4   output en,
5   output reg [7:0] count
6 );
7 always @(posedge clk) begin
8   if (!rst_n)
9     count <= 8'd0;
10  else if (en)
11    count <= count + 1;
12  else
13    count <= count; // Hold value
14 end
endmodule
    
```



Additional AI Suggestions

- Refactor for better readability
- Optimize for area/power (if needed)
- Add assertions for corner cases
- Generate additional test cases
- Improve reset/enable handling

5 VALIDATION & RESULTS

- Compilation Successful
- All Assertions Passed
- Refactoring
- Test Case Passed
- Waveform Verified
- Coverage Improved

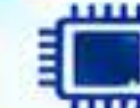
Waveform (Fixed)



Coverage Improvement



- Code Coverage ↑
- Assertion Coverage ↑
- Functional Coverage ↑





INPUTS (Multi-Source)



SiliconAI
RTL Intelligence Engine
(Multi-Agent AI + Knowledge Graph)

ACTIONABLE OUTPUTS



1 RTL Code Intelligence Deep code analysis beyond linting

- ✓ Coding errors & synthesis risks
- ✓ Unsynthesizable constructs
- ✓ Race conditions & latch inference
- ✓ Reset & clock-domain issues
- ✓ Width/sign mismatches
- ✓ Dead code & unreachable states
- ✓ Blocking/non-blocking misuse
- ✓ Suspicious conditions & parameters

```
1 module counter (
2   input clk, reset,
3   input enable,
4   output reg [7:0] count
5 );
6
7 always @(posedge clk) begin
8   if (enable)
9     count <= count + 1;
10 end
11 endmodule
```

AI Analysis

- Unreachable state
- Latch inference
- Width mismatch
- Missing reset
- Recommended fix

Key Output

Critical issues | Severity | Root cause | Recommended fix

2 RTL Debug Intelligence From Failure to Fix with AI

- ✓ Analyze simulation failures
- ✓ Understand logs, waveforms, assertions
- ✓ Trace signal & state behavior
- ✓ Map to RTL line & specification
- ✓ Identify root cause & propose fix
- ✓ Generate regression test & verify



Key Output

What failed → Where → When → Why → Which RTL → What to change → How to verify
(Failure → Root Cause → Fix → Proof)

3 Waveform Intelligence AI-powered waveform analysis

- ✓ Find first divergence
- ✓ Detect unexpected/missing transitions
- ✓ Identify glitches & protocol violations
- ✓ Analyze timing & state anomalies
- ✓ Trace back to RTL and root cause
- ✓ Visual comparison (expected vs actual)



Key Output

Waveform anomaly → Signal trace → RTL line → Root cause → Recommended fix

4 Coverage Intelligence Beyond coverage numbers

- ✓ Find uncovered branches, states, transitions
- ✓ Identify unreachable or dead code
- ✓ Distinguish testbench issue vs RTL issue
- ✓ Generate targeted test cases
- ✓ Run simulation and measure improvement
- ✓ Coverage-to-root-cause analysis



Key Output

Why coverage is low → Coverage holes → Root cause → Generated tests → Improved coverage

5 Critical RTL Issue Intelligence RTL Health Score & Risk Analysis

- ✓ Functional, verification & CDC risks
- ✓ Reset, interface, complexity analysis
- ✓ Top 10 critical issues per module
- ✓ Risk scoring and recommendations
- ✓ SoC-level risk heatmap
- ✓ Prioritize fixes for engineering teams



Key Output

Module risk score | Top issues | Prioritized fixes
SoC risk dashboard

6 RTL → Specification Reverse Engineering

Generate specification from existing RTL

- ✓ Understand architecture, interfaces
- ✓ Extract FSM, registers, datapath, control
- ✓ Infer timing behavior & requirements
- ✓ Generate human-readable specification
- ✓ Useful for legacy IP and maintenance

AI Reconstructed Specification

Module: DMA Controller

Inputs:
• clk, reset, start, src_addr, dst_addr, length

Behavior:

1. Accept start request
2. Validate address
3. Initialize transfer
4. Generate read requests
5. Generate write requests
6. Increment address
7. Assert done

Key Output

Module architecture → Interfaces → FSM → Behavior → Generated specification (DOC)

7 Specification ↔ RTL Traceability End-to-end requirement tracking

- ✓ Map requirements to RTL, assertions, tests
- ✓ Track verification status for each requirement
- ✓ Identify verification gaps
- ✓ Bidirectional traceability
- ✓ Compliance and reporting

Requirement Traceability

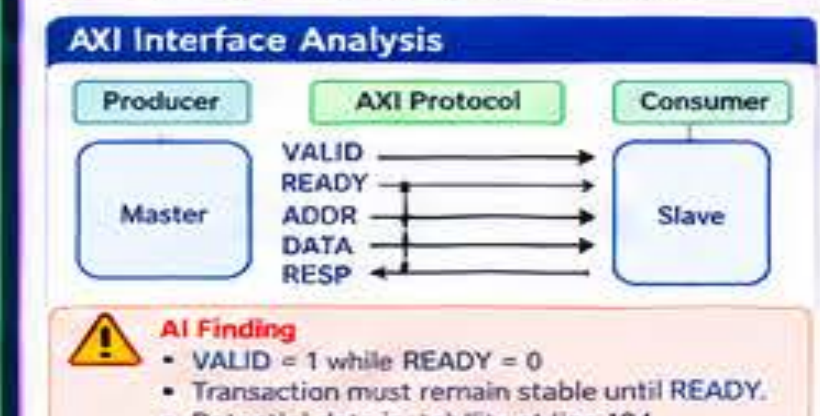
ID	Requirement	RTL	Assert	Test	Coverage	Status
R-127	DMA burst support	✓	✓	✓	94%	Verified
R-214	ECC error handling	✓	✗	✗	0%	Gap
R-221	Low power mode	✓	✓	✓	76%	In Progress
R-305	Timeout detection	✓	✓	✓	92%	Verified

Key Output

Requirement → RTL → Assertion → Test → Coverage → Verification status

8 Interface Intelligence Protocol-aware RTL analysis

- ✓ Understand standard protocols (AXI, AHB, APB, PCIe, DDR, USB, Ethernet, custom)
- ✓ Detect protocol violations & handshake issues
- ✓ Check width, ordering, burst, timeout, backpressure
- ✓ Analyze reset interaction and assumptions

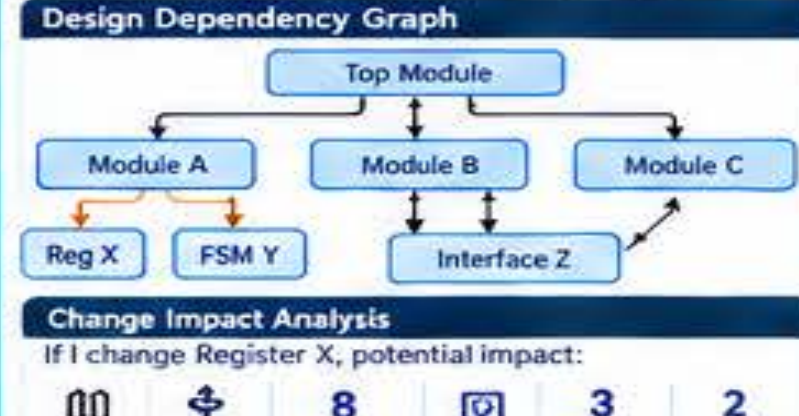


Key Output

Protocol violations → Interface issues → Root cause → Recommended fix

9 RTL Dependency Intelligence Design dependency & impact analysis

- ✓ Build module and signal dependency graph
- ✓ Trace register/logic/module dependencies
- ✓ Analyze impact of code changes
- ✓ Find affected assertions, tests, interfaces
- ✓ Predict risk before committing changes



Key Output

Dependency graph → Impact analysis → Risk prediction → Recommended regression

10 AI Refactoring & Optimization Better RTL with PPA and equivalence

- ✓ Analyze current RTL and behavior
- ✓ Generate optimized alternatives
- ✓ Check formal/simulation equivalence
- ✓ Estimate PPA (Area, Power, Performance)
- ✓ Recommend best implementation

Refactoring Alternatives

Current RTL	Alternative A (Recommended)	Alternative B
Area: 100K Power: 50mW Perf: 1.0x Risk: -	Area: 92K (-8%) Power: 48mW (-4%) Perf: 1.02x (-2%) Risk: Low ✓	Area: 85K (-15%) Power: 44mW (-11%) Perf: 0.97x (-3%) Risk: Medium

AI Recommendation
Alternative A — better balanced PPA with lower functional risk.

Key Output

Optimized RTL → PPA estimation → Equivalence check → Best implementation

11 Legacy RTL Archaeology Recover design history & intent

- ✓ Analyze old RTL, docs, bugs and comments
- ✓ Reconstruct design intent and evolution
- ✓ Link historical bugs, fixes, and engineers
- ✓ Explain unusual logic and design decisions
- ✓ Useful for IP acquisition and maintenance



Key Output

Design evolution → Historical context → Intent explanation → Knowledge retrieval

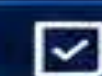
12 RTL Knowledge Graph Central intelligence layer

- ✓ Connect requirements, architecture, RTL, tests, assertions, coverage and bugs
- ✓ Continuously learn from new projects
- ✓ Enable natural language queries
- ✓ Build company-wide engineering intelligence



Key Output

Unified knowledge graph → Search & query → Reuse → Better future designs





1

RTL Code Intelligence

Go Beyond Linting — Deep AI Analysis of RTL Code

Find Issues. Understand Intent.
Explain Root Cause. Fix with AI.

RTL Input



SiliconAI Code Intelligence Engine

Parse • Understand • Analyze
Build Design Model (AST + CFG + FSM)
Apply AI Analysis Agents

Design Understanding



Deep Analysis (Beyond Linting)

- ✓ Coding errors
- ✓ Synthesis risks
- ✓ Unsynthesizable constructs
- ✓ Race conditions
- ✓ Incomplete assignments
- ✓ Latch inference
- ✓ Blocking/non-blocking misuse
- ✓ Reset problems
- ✓ Clock-domain issues (CDC)
- ✓ Width/sign mismatches
- ✓ Dead code
- ✓ Duplicate logic
- ✓ Unreachable states
- ✓ Suspicious conditions
- ✓ Parameter problems

Example: AI Analysis of RTL Code

counter.v

```
1 module counter (
2     input clk,
3     input rst_n,
4     input enable,
5     output reg [7:0] count
6 );
7
8 always @(posedge clk) begin
9     if (enable)
10        count <= count + 1;
11 end
12 endmodule
```

- Latch inference (count not assigned in else)
- Missing reset handling
- Width mismatch (count may overflow)
- Blocking/Non-blocking issue
- Unreachable state
- Parameter mismatch

How SiliconAI Works

- 1 Understand RTL Structure (Parse, AST, CFG, FSM)
- 2 Analyze & Detect Issues (Rules + AI Reasoning)
- 3 Classify Severity (Critical / High / Medium / Low)
- 4 Find Root Cause (Explain with context)
- 5 Recommend Fix (With code suggestion)

AI Analysis Results

#	Issue	Severity	Root Cause	Recommendation
1	Latch inference	Critical	count not assigned in else	Add else assignment or default value
2	Missing reset	High	No reset for count register	Add reset logic
3	Width mismatch	High	count may overflow	Use proper width or saturation logic
4	Blocking assignment	Medium	= used in sequential logic	Use <= for sequential logic
5	Unreachable state	Medium	FSM state S3 never reached	Check state transition conditions
6	Unused signal	Low	temp_sig is never used	Remove unused signal
7	Parameter issue	Low	WIDTH not defined	Define parameter or use default

AI Suggested Fix

Original Code

```
always @(posedge clk) begin
    if (enable)
        count <= count + 1;
end
```

Fixed Code

```
always @(posedge clk) begin
    if (!rst_n)
        count <= 8'd0;
    else if (enable)
        count <= count + 1;
    else
        count <= count; // hold
end
```

2

RTL Debug Intelligence

From Simulation Failure to Verified Fix with AI

What failed? Where? When? Why?
Which RTL caused it? What should change?
How to verify the fix?

Multi-Source Inputs



SiliconAI Debug Intelligence Engine

Multi-Agent AI Analysis
Logs • Waveforms • RTL • Assertions
Specification • Coverage

Example: Debug Session

Simulation Log (Error)

```
# ** Error: counter_tb.sv(45):
# Assertion failed: count should
# not increment when en=0
# Time: 250 ns
# * Fatal: Simulation stopped.
#
```

Waveform (VCD/FSDB)



AI Waveform Analysis



Signal to RTL Trace

- Signal: count
 - Changed when en = 0
 - Traced to:
- ```
counter.v:8
if (enable)
 count <= count + 1;
```

### AI Debug Result — Failure → Root Cause → Fix → Proof

- |                                                                               |                                                          |                                                                        |                                                                            |                                                        |                                                                                     |
|-------------------------------------------------------------------------------|----------------------------------------------------------|------------------------------------------------------------------------|----------------------------------------------------------------------------|--------------------------------------------------------|-------------------------------------------------------------------------------------|
| <b>1 What Failed</b><br>Count incremented when enable = 0 (Assertion failure) | <b>2 Where &amp; When</b><br>counter.v:8<br>Time: 250 ns | <b>3 Why</b><br>Missing else condition. count not held when enable = 0 | <b>4 Root Cause</b><br>Incomplete logic count updated even when enable = 0 | <b>5 Proposed Fix</b><br>Add else branch to hold value | <b>Verification</b><br>Run regression test<br>All tests passed<br>Coverage improved |
|-------------------------------------------------------------------------------|----------------------------------------------------------|------------------------------------------------------------------------|----------------------------------------------------------------------------|--------------------------------------------------------|-------------------------------------------------------------------------------------|

### Key Outputs



### Key Benefits





3

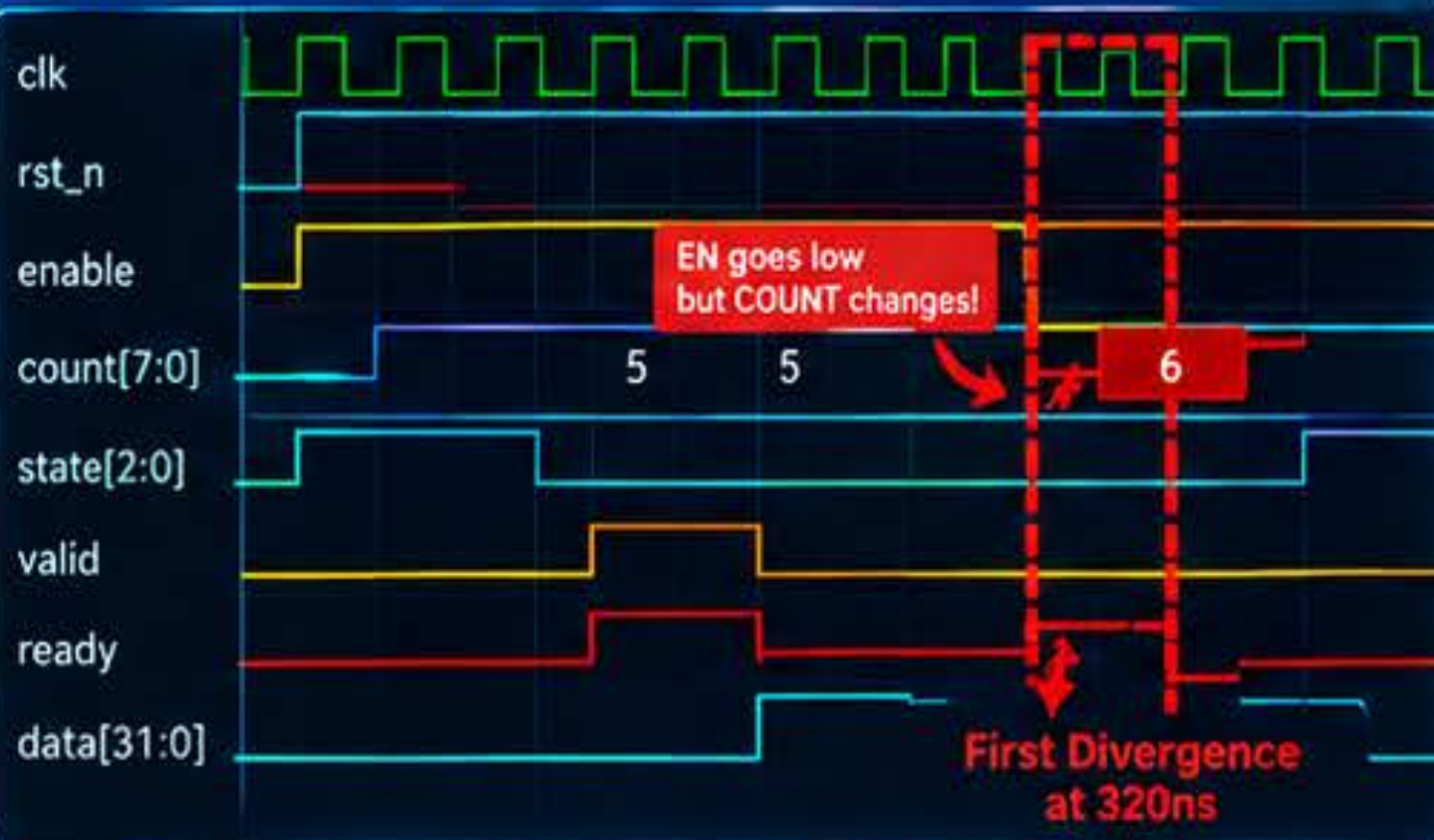
## Waveform Intelligence

AI-Powered Waveform Analysis — Automatically Find Anomalies & Root Cause

From Waveforms to Root Cause  
in Minutes, Not Hours.



### Waveform Analysis with AI Insights



### Expected vs Actual (Zoomed View)



### AI Analysis & Root Cause Trace

- Waveform Anomaly Detected  
Count changed when enable = 0
- Signal Dependency Trace  
Traced to count register update
- RTL Line Trace  
Identified in counter.v line 42
- Condition Analysis  
Missing enable check in update logic
- Root Cause  
Count increments even when enable = 0
- Recommended Fix  
Add enable condition and verify

### RTL Code (counter.v)

```
38 always @(posedge clk) begin
39 if (rst_n) begin
40 if (enable)
41 count <= count + 1;
42 else
43 count <= count; // Hold value
44 end
45 end
```

Missing enable condition  
or incorrect logic

### AI Suggested Fix

```
always @(posedge clk) begin
 if (rst_n) begin
 if (enable)
 count <= count + 1;
 else
 count <= count; // Hold value
 end
end
```

- Fixes first divergence
- Matches spec behavior
- Synthesizable
- No new warnings
- Run regression test

4

## Coverage Intelligence

Beyond Numbers — Find Why Coverage is Low & How to Improve

From Coverage Gaps  
to Better Tests & Higher Quality.



### Coverage Dashboard



### Coverage Breakdown by Module

| Module   | Line | Branch | FSM | Assertion | Status   |
|----------|------|--------|-----|-----------|----------|
| uart_rx  | 92%  | 88%    | 84% | 81%       | Good     |
| uart_tx  | 76%  | 62%    | 48% | 51%       | Low      |
| dma_ctrl | 68%  | 54%    | 32% | 40%       | Critical |
| axi_if   | 85%  | 70%    | 67% | 64%       | Low      |
| fifo     | 95%  | 90%    | 88% | 82%       | Good     |
| arbiter  | 60%  | 42%    | 35% | 38%       | Critical |

### Uncovered FSM States Analysis (dma\_ctrl)



### Why Uncovered?

#### Possible Causes (AI Analysis)

- Missing test case to trigger write state
- Testbench does not generate this scenario
- State is hard to reach due to conditions
- Possible design bug (unreachable)
- Requirement not properly tested

### Coverage-to-Root-Cause Analysis Flow

- Coverage Gap  
Uncovered state/branch/transition
- Why Unreachable?  
Analyze RTL, tests, spec & constraints
- Identify Cause  
Missing test / Dead RTL / Impossible state / Missing requirement / Design bug
- Generate Test  
Create targeted test case using AI
- Run Simulation  
Execute and collect coverage
- Improve Coverage  
Measure improvement and validate

### AI Suggested Test Case

```
// Test to trigger write state
initial begin
 reset();
 start_transfer(src_addr, dst_addr, length);
 wait_for_write_state();
 check_write_behavior();
end
```

### Coverage Improvement (After AI Test)



### Key Outputs

- Uncovered items identified
- Root cause analysis
- AI-generated test cases
- Improved coverage results
- Verification closure guidance

### KEY BENEFITS



Find bugs faster  
with AI waveform analysis



Deep root cause  
explanation



Higher coverage  
with AI guidance



Reduced debug time  
60% - 80%



Better verification quality  
and confidence



CLEAN RTL. HIGHER COVERAGE.  
FASTER DEBUG. BETTER SILICON.



5

Critical RTL Issue Intelligence  
AI-Powered RTL Risk Analysis & Health ScoreTurn Thousands of Warnings  
into Actionable InsightsCode Quality  
AnalysisFunctional  
RiskVerification  
RiskInterface  
AnalysisCDC & Reset  
AnalysisCoverage  
AnalysisComplexity &  
MaintainabilityPPA  
Estimation

## Module RTL Health Score

dma\_ctrl



| Risk Category     | Score |
|-------------------|-------|
| Functional Risk   | 82    |
| Verification Risk | 71    |
| Complexity        | 64    |
| Maintainability   | 58    |
| Interface Risk    | 81    |
| CDC Risk          | 35    |
| Reset Risk        | 76    |
| Coverage Risk     | 69    |
| PPA Risk          | 62    |

Overall RTL Risk: HIGH

Requires immediate attention and mitigation

## SoC Module Risk Overview

| Module     | Health Score | Risk Level | Top Issues           |
|------------|--------------|------------|----------------------|
| dma_ctrl   | 68           | HIGH       | 5 critical issues    |
| axi_master | 82           | MEDIUM     | CDC, interface       |
| uart       | 88           | LOW        | No major issues      |
| i2c_ctrl   | 61           | HIGH       | Reset, coverage      |
| spi_ctrl   | 76           | MEDIUM     | FSM, corner cases    |
| mem_ctrl   | 55           | HIGH       | Timing, CDC          |
| clk_gen    | 90           | LOW        | Clean                |
| rst_sync   | 48           | HIGH       | Reset, metastability |
| top_soc    | 72           | MEDIUM     | Integration issues   |

## Top 10 Most Important RTL Issues (AI Prioritized)

| #  | Module     | Issue                  | Severity | Root Cause              | Recommended Fix        |
|----|------------|------------------------|----------|-------------------------|------------------------|
| 1  | dma_ctrl   | Latch inference        | Critical | Incomplete assignment   | Add else assignment    |
| 2  | mem_ctrl   | CDC violation          | Critical | Async signal crossing   | Add synchronizer       |
| 3  | rst_sync   | Reset not asserted     | Critical | Improper reset logic    | Fix reset sequence     |
| 4  | i2c_ctrl   | FSM unreachable        | High     | Missing transition      | Add transition logic   |
| 5  | dma_ctrl   | Data width mismatch    | High     | Width/sign mismatch     | Fix port width         |
| 6  | axi_master | Protocol violation     | High     | AXI handshake issue     | Fix valid/ready logic  |
| 7  | spi_ctrl   | Dead code              | Medium   | Unused logic            | Remove unused logic    |
| 8  | dma_ctrl   | Timeout risk           | Medium   | No timeout handling     | Add timeout counter    |
| 9  | uart       | Unreachable state      | Medium   | FSM state never reached | Check state conditions |
| 10 | mem_ctrl   | Performance bottleneck | Low      | Inefficient logic       | Optimize logic         |

## Risk Distribution



## Risk by Category (All Modules)

|                 |    |
|-----------------|----|
| Functional      | 72 |
| Verification    | 65 |
| Complexity      | 58 |
| Maintainability | 61 |
| Interface       | 68 |
| CDC             | 49 |
| Reset           | 63 |
| Coverage        | 60 |
| PPA             | 55 |

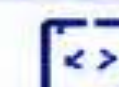
## Risk Trend (Over Time)



## Key Benefits for Engineering Teams

- Identify critical issues faster
- Prioritize by real impact
- Reduce debug time 40% - 70%
- Improve RTL quality
- Management-level visibility
- Better PPA, higher reliability

6

RTL → Specification Reverse Engineering  
AI Reconstructs Design Intent, Architecture and Specification from Existing RTLTurn Legacy RTL  
into Clear, Usable  
DocumentationExisting RTL  
(10+ Years Old)Module  
ArchitectureInterfaces  
(Protocols & Ports)FSM  
(State Machines)Registers  
& SignalsData Path &  
Control PathTiming  
BehaviorAI-Generated  
Specification

## Input: Legacy RTL Code (Example: dma\_ctrl)

```
1 module dma_ctrl (
2 input clk,
3 input rst_n,
4 input start,
5 input [31:0] src_addr,
6 input [31:0] dst_addr,
7 input [15:0] length,
8 output reg done,
9
10 // ... legacy code ...
11 always @(posedge clk or negedge rst_n) begin
12 if (!rst_n) begin
13 // reset logic
14 end else begin
15 // control logic
16 end
17 end
endmodule
```

## AI Analysis Pipeline

- 1 Parse & Understand RTL  
Extract modules, signals, structure
- 2 Reconstruct Architecture  
Identify data path and control path
- 3 Identify Interfaces  
Protocols, ports, handshakes
- 4 Extract FSM  
States, transitions, conditions
- 5 Analyze Registers  
Control, status, configuration
- 6 Infer Timing Behavior  
Latencies, sequences, constraints
- 7 Derive Implicit Requirements  
From code behavior & patterns
- 8 Generate Specification  
Functional, interface, timing, behavior

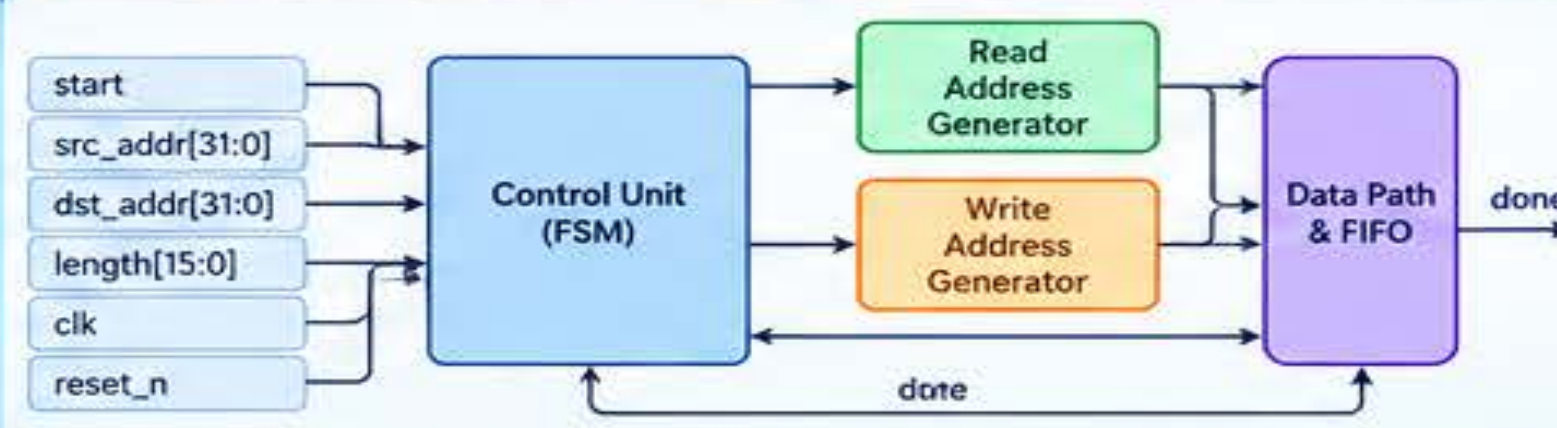
## Output: AI-Reconstructed Specification

Specification Block Diagram FSM Diagram

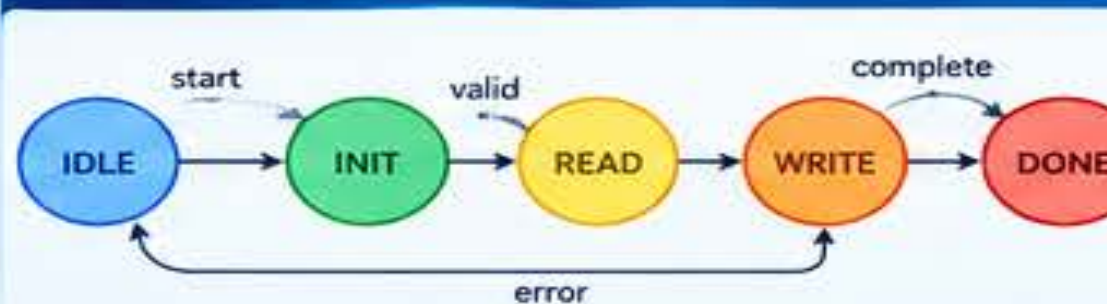
## Module: DMA Controller

1. Overview  
DMA controller transfers data from source to destination memory using configurable length. Supports burst transfers and generates done signal upon completion.
2. Inputs  
clk : Clock  
reset : Active low reset  
start : Start DMA transfer  
src\_addr : Source address (32-bit)  
dst\_addr : Destination address (32-bit)  
length : Transfer length (16-bit)
3. Outputs  
done : High when transfer is complete
4. Functional Behavior  
1. Accept start request  
2. Validate address  
3. Initialize transfer  
4. Generate read requests  
5. Generate write requests  
6. Increment address  
7. Assert done
5. Timing Behavior  
• Read latency: 2 cycles  
• Write latency: 2 cycles  
• Supports burst transfers  
• Timeout if no response (configurable)

## Reconstructed Architecture (Block Diagram)



## FSM Reconstruction



## Interface Analysis

| Interface | Type    | Direction | Width | Description                |
|-----------|---------|-----------|-------|----------------------------|
| clk       | Clock   | Input     | 1     | System clock               |
| reset_n   | Reset   | Input     | 1     | Active low reset           |
| start     | Control | Input     | 1     | Start transfer             |
| src_addr  | Address | Input     | 32    | Source memory address      |
| dst_addr  | Address | Input     | 32    | Destination memory address |
| length    | Control | Input     | 16    | Transfer length            |
| done      | Status  | Output    | 1     | Transfer complete          |

## Key Benefits for Legacy IP and Maintenance

- Understand old RTL quickly
- Generate clear documentation
- Reduce engineer onboarding time
- Preserve design knowledge
- Enable IP reuse and integration
- Better verification and maintenance



## Traceability Flow (Two-Way Mapping)



## SPECIFICATION

- Requirements
- Functional Description
- Use Cases



## ARCHITECTURE

- System Design
- Module Decomposition
- Interface Definition



## RTL

- Module Implementation
- Signals & Registers
- FSM & Datapath



## ASSERTIONS

- SVA/PSL Assertions
- Protocol Checks
- Functional Properties



## TEST CASE

- Directed / Random Tests
- Constrained Random
- Scenario Based Tests



## WAVEFORM

- Simulation Results
- Signal Transitions
- Debug Analysis



## COVERAGE

- Code Coverage
- Functional Coverage
- Assertion Coverage



## AI-Powered Traceability Engine

Parse & Understand  
Specs, RTL, TestsMap & Link  
AI MatchingAnalyze Gaps  
Missing LinksValidate  
Run & VerifyVisualize  
Interactive Traceability

## Requirement Traceability Matrix (AI Generated)

Search requirement, module, signal...

| ID    | Requirement / Specification   | Architecture | RTL Module   | Assertion               | Test Case             | Simulation | Coverage | Status           | Risk   |
|-------|-------------------------------|--------------|--------------|-------------------------|-----------------------|------------|----------|------------------|--------|
| R-125 | Support burst transfer of 4K  | AXI Master   | axi_master.v | axi_burst_assert.sv     | test_burst_4k.sv      | ✓          | 96%      | VERIFIED         | Low    |
| R-126 | Handle read and write request | AXI Master   | axi_master.v | axi_rw_assert.sv        | test_read_write.sv    | ✓          | 92%      | VERIFIED         | Low    |
| R-127 | DMA transfer completion       | DMA Ctrl     | dma_ctrl.v   | dma_done_assert.sv      | test_dma_done.sv      | ✓          | 94%      | VERIFIED         | Low    |
| R-128 | Address alignment check       | DMA Ctrl     | dma_ctrl.v   | addr_align_assert.sv    | test_addr_align.sv    | ✓          | 90%      | VERIFIED         | Medium |
| R-129 | Error handling and retry      | Error Ctrl   | err_ctrl.v   | err_retry_assert.sv     | test_error_retry.sv   | ✓          | 88%      | VERIFIED         | Medium |
| R-130 | FIFO overflow protection      | FIFO         | fifo.v       | fifo_overflow_assert.sv | test_fifo_overflow.sv | ✓          | 87%      | VERIFIED         | Medium |
| R-214 | Low power mode support        | Power Mgmt   | pm_ctrl.v    | ↔                       | ↔                     | ✗          | 0%       | VERIFICATION GAP | High   |
| R-215 | Clock domain synchronization  | CDC          | cdc_sync.v   | ↔                       | ↔                     | ✗          | 0%       | VERIFICATION GAP | High   |
| R-216 | Reset sequence handling       | Reset Ctrl   | rst_ctrl.v   | rst_seq_assert.sv       | test_reset_seq.sv     | ✓          | 78%      | PARTIAL          | Medium |
| R-217 | Interrupt generation          | IRQ Ctrl     | irq_ctrl.v   | irq_assert.sv           | test_interrupt.sv     | ✓          | 85%      | VERIFIED         | Low    |

## Example 1: Verified Requirement (R-127)

## Specification

R-127: Assert done when  
DMA transfer is complete.

## RTL (dma\_ctrl.v)

```
120 if (transfer_complete) begin
121 done <= 1'b1;
122 end
```

## Assertion

```
property done_assert;
 @(posedge clk)
 transfer_complete |
 => ##1 done;
endproperty
```

## Test Case

```
test_dma_done.sv
- Initialize transfer
- Wait for completion
- Check done signal
```

## Waveform (VCD)

clk  
transfer\_complete  
done

Done asserted  
at expected time

## Coverage

94%

Functional Coverage | Assertion Coverage | Code Coverage

Status: VERIFIED

## Example 2: Verification Gap (R-214)

## Specification

R-214: Support low power  
mode with clock gating.

## RTL (pm\_ctrl.v)

```
45 if (low_power_en) begin
46 // clock gating logic
47 end
```

## Assertion

No assertion found

## Test Case

No test case found

## Waveform

No relevant waveform  
found

## Coverage

0%  
No functional coverage

Status: VERIFICATION GAP

## Requirement Details &amp; Trace Links (R-127)

✓ Verified

## R-127 DMA transfer completion

Priority:

High

Category:

Functional

Description:

DMA controller shall assert 'done' signal  
when transfer is complete.

Source:

system\_spec\_v1.2.docx

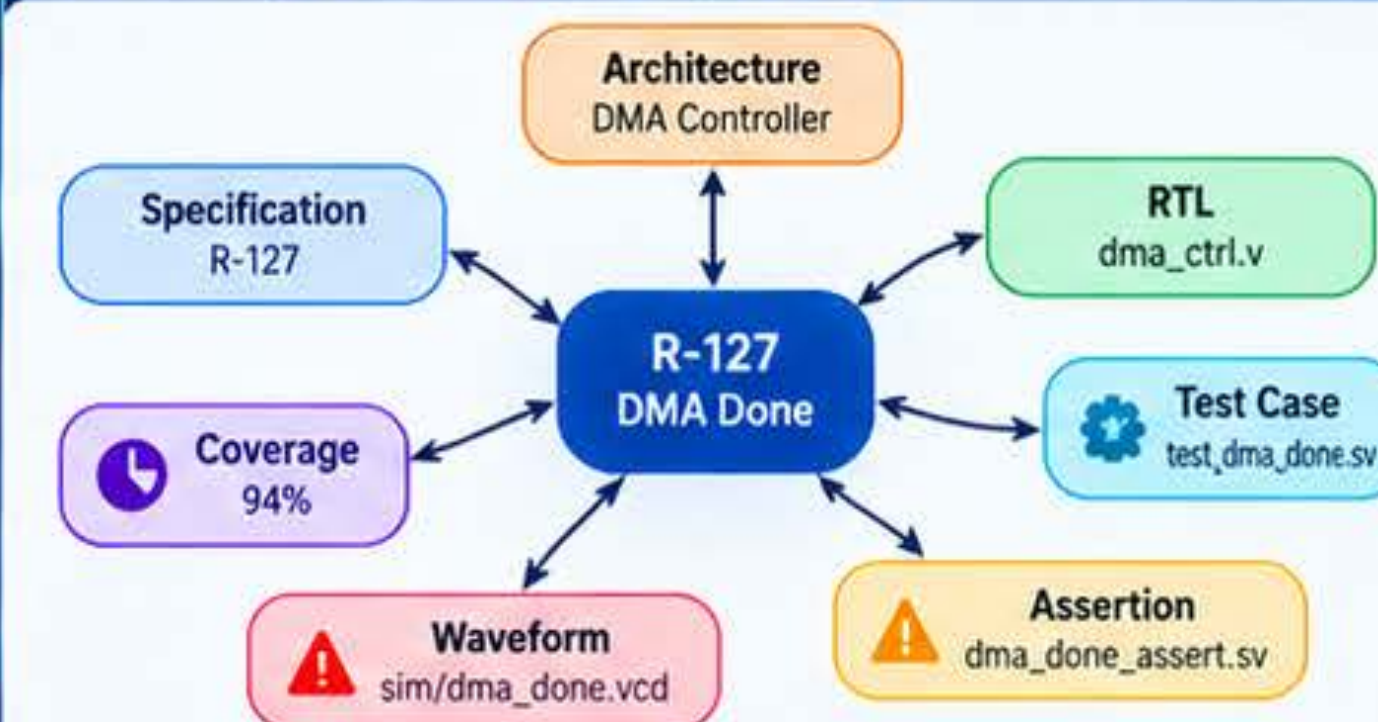
Coverage

94%

## + Trace Links

|                    |                              |                        |   |
|--------------------|------------------------------|------------------------|---|
| Architecture       | DMA Controller               | arch/dma_controller.md | ✓ |
| RTL Implementation | dma_ctrl.v (lines 120-180)   |                        | ✓ |
| Assertion          | dma_done_assert.sv           |                        | ✓ |
| Test Case          | test_dma_done.sv             |                        | ✓ |
| Waveform           | sim_results/dma_done.vcd     |                        | ✓ |
| Coverage           | functional_cov/dma_ctrl.ucdb |                        | ✓ |

## Traceability Visualization (Graph View)



## Key Benefits

- Complete Traceability
- Faster Debug & Verification
- Find Gaps Automatically
- Better Compliance
- Better Compliance
- Reuse & Maintainability
- Reuse & Maintainability
- Lower Risk Silicon
- AI-Powered Insights



### Design Hierarchy (Module View)



### Selected Register Details

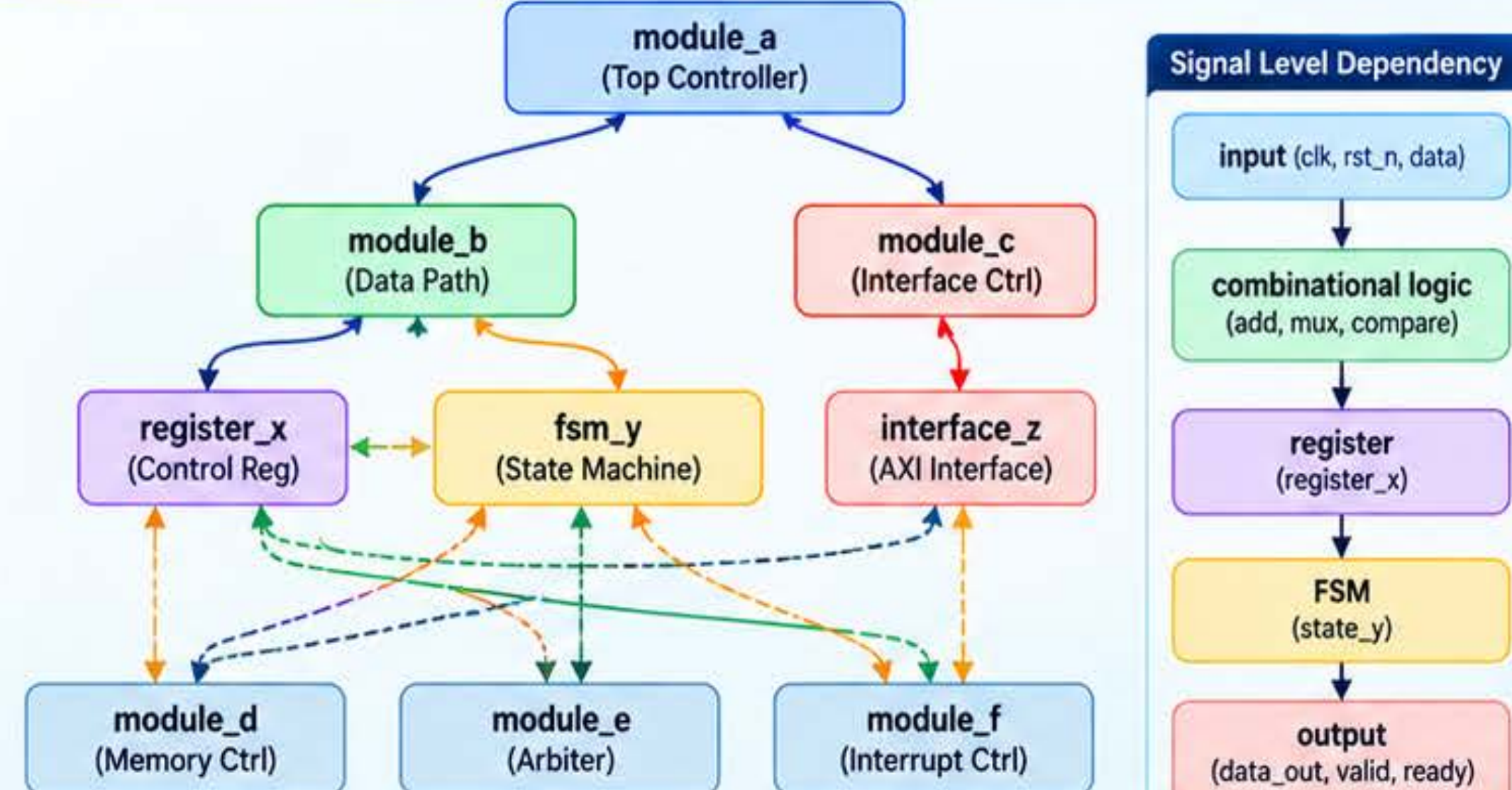
Module: module\_b Register: register\_x

```
45 always @(posedge clk or negedge rst_n) begin
46 if (!rst_n)
47 register_x <= 1'b0;
48 else if (enable)
49 register_x <= data_in;
50 end
```

### Register Usage (Who Reads/Writes)

| Written By (3)               | Read By (6)                    |
|------------------------------|--------------------------------|
| 1 module_a/control_fsm.v:120 | 1 module_b/data_path.v:210     |
| 2 module_d/mem_ctrl.v:88     | 2 module_c/axi_interface.v:134 |
| 3 module_e/arbitrator.v:45   | 3 module_f/interrupt.v:76      |
|                              | ... 3 more readers             |

### RTL Dependency Graph (Interactive)



### Dependency Details

| Type      | Name             | Relationship            | Details                         | Impact |
|-----------|------------------|-------------------------|---------------------------------|--------|
| Module    | module_c         | Reads register_x        | Affects AXI transaction control | High   |
| Module    | module_d         | Depends on output       | Impacts memory address gen      | High   |
| Module    | module_f         | Uses status signal      | Interrupt generation            | Medium |
| Signal    | data_valid       | Driven by register_x    | Affects handshake               | High   |
| Signal    | addr_out[31:0]   | Derived from register_x | Impacts memory access           | High   |
| Assertion | axi_valid_stable | Depends on register_x   | May fail with new logic         | Medium |
| Assertion | reset_sequence   | Indirectly affected     | Reset behavior change           | Medium |
| Test Case | test_axi_write   | Stimulates register_x   | Verify write sequence           | High   |
| Test Case | test_mem_access  | Uses output signal      | Memory transaction test         | Medium |
| Interface | AXI4             | Transaction affected    | VALID/READY timing              | High   |
| Coverage  | axi_write_cg     | Bins may change         | Coverage drop possible          | Medium |

### Change Impact Analysis



If I change register\_x in module\_b, what could break?



Analyze Impact



### AI Analysis Result

Risk Level: **HIGH**

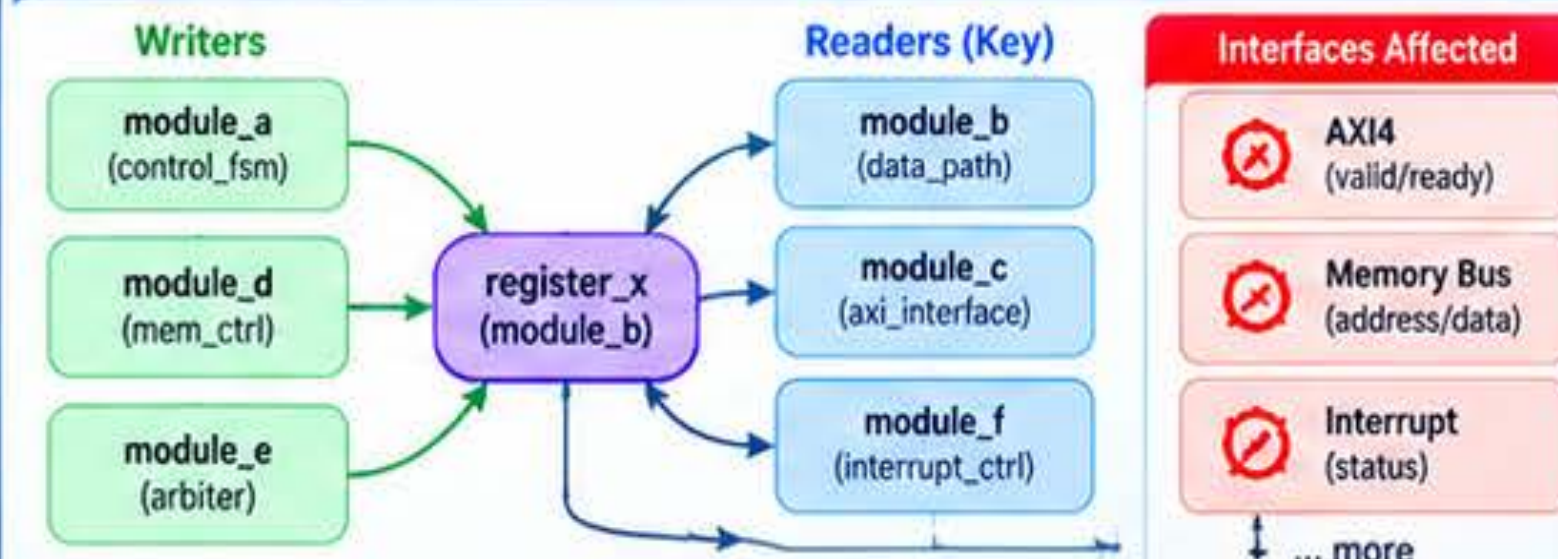
#### Potential Impact

|    |                          |
|----|--------------------------|
| 12 | Modules affected         |
| 47 | Signals affected         |
| 8  | Assertions affected      |
| 23 | Test cases affected      |
| 3  | Interfaces affected      |
| 2  | Coverage groups affected |

#### Affected Modules (12)

|     |                           |
|-----|---------------------------|
| 1   | module_a (Top Controller) |
| 2   | module_b (Data Path)      |
| 3   | module_c (Interface Ctrl) |
| 4   | module_d (Memory Ctrl)    |
| 5   | module_e (Arbiter)        |
| 6   | module_f (Interrupt Ctrl) |
| 7   | axi_master (IP)           |
| 8   | mem_ctrl (IP)             |
| ... | 4 more modules            |

### Impact Visualization (Fan-out/Fan-in)



### Recommendations

- Run affected test cases (23) after modification
- Check AXI VALID/READY timing
- Verify reset behavior and initialization
- Update related assertions if required
- Re-run coverage to ensure no drop
- Consider formal equivalence check for safety



### Change Risk Summary

Change Type: Register Logic Modification  
Risk Level: **HIGH**  
Potential Bugs: Protocol violation, timing, FSM  
Suggested Actions:

- Run full regression on affected modules
- Check assertions and coverage
- Review interface timing
- Perform formal check (if available)



### Current RTL (Original Code)

Before

```
// Counter with enable and reset
module counter_old (
 input wire clk,
 input wire rst_n,
 input wire enable,
 input wire [7:0] data_in,
 output reg [7:0] count
);
 always @(posedge clk or negedge rst_n) begin
 if (!rst_n) begin
 count <= 8'd0;
 end else begin
 if (enable) begin
 count <= count + data_in;
 end
 end
 end
endmodule
```

### AI Analysis

- Understand design intent
- Analyze control & datapath
- Identify optimization opportunities
- Detect redundant logic
- Analyze critical path
- Check reset and enable logic

### Optimization Opportunities

- Reduce register and logic count
- Share common logic
- Improve critical path
- Reduce switching activity
- Better reset structure
- Parameterization for reuse

### Alternative A (Recommended)

Optimized & Balanced

```
// Optimized counter - shared logic, cleaner structure
module counter_opt_a (
 input wire clk,
 input wire rst_n,
 input wire enable,
 input wire [7:0] data_in,
 output reg [7:0] count
);
 wire [7:0] next_count = count + data_in;

 always @(posedge clk) begin
 if (!rst_n)
 count <= 8'd0;
 else if (enable)
 count <= next_count;
 end
endmodule
```

### Alternative B (High Optimization)

More Aggressive

```
// High optimization - pipelined and gated logic
module counter_opt_b (
 input wire clk,
 input wire rst_n,
 input wire enable,
 input wire [7:0] data_in,
 output reg [7:0] count
);
 wire [7:0] add_result;
 assign add_result = enable ? (count + data_in) : count;

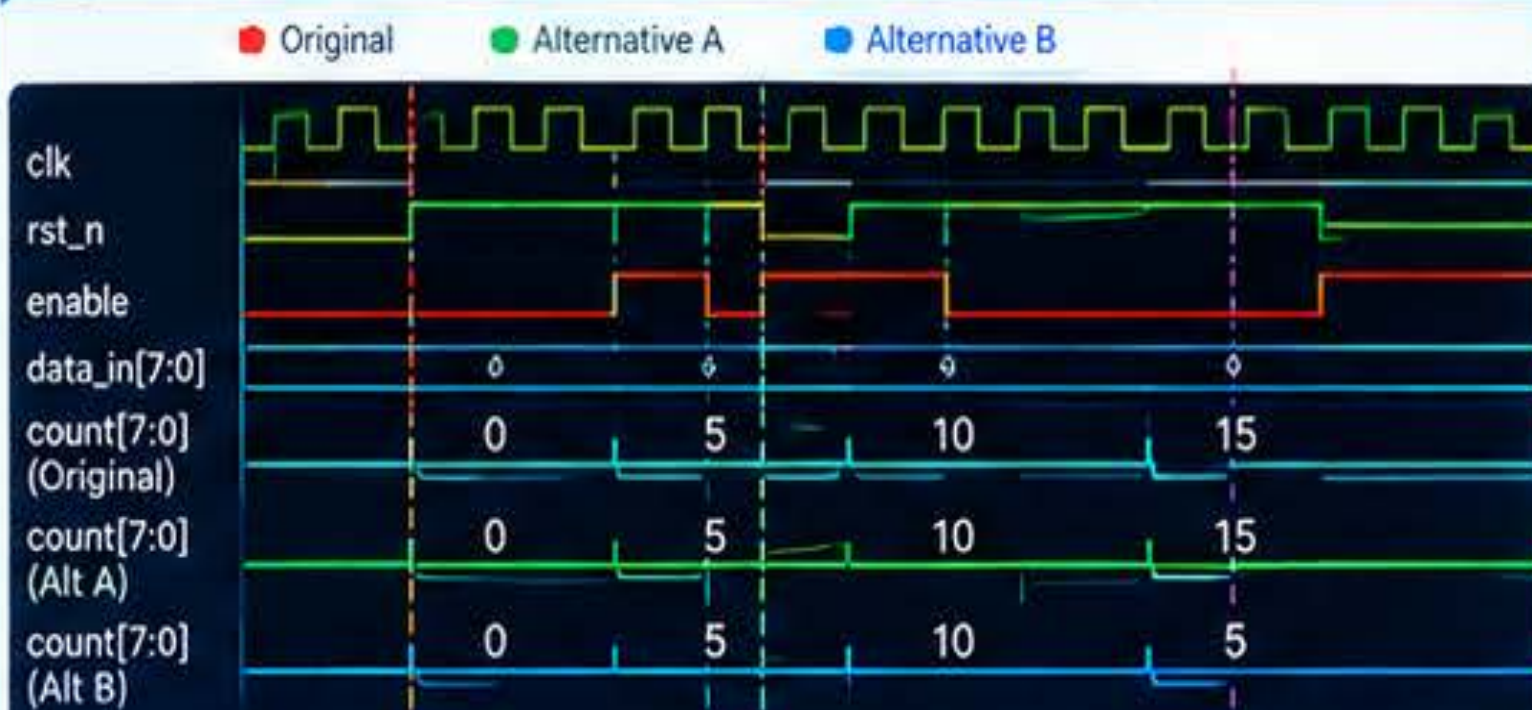
 always @(posedge clk) begin
 if (!rst_n)
 count <= 8'd0;
 else
 count <= add_result;
 end
endmodule
```

### Functional Equivalence Verification

- Formal Verification **PASSED**
- Simulation (Testbench) **PASSED**
- Constrained Random **PASSED**
- Corner Cases **PASSED**
- Reset Behavior **PASSED**
- Coverage Check **PASSED**

### Simulation Waveform Comparison

Waveforms Match ✓



### PPA Comparison (Post-Synthesis)

| Metric                 | Original | Alternative A | Alternative B |
|------------------------|----------|---------------|---------------|
| Area (µm²)             | 12,500   | 11,500 (-8%)  | 10,600 (-15%) |
| Power (mW)             | 2.50     | 2.40 (-4%)    | 2.22 (-11%)   |
| Performance (Max Freq) | 500 MHz  | 510 MHz (+2%) | 485 MHz (-3%) |
| Latency (cycles)       | 1        | 1             | 1             |
| Logic Levels           | 4        | 3             | 3             |
| Registers              | 8        | 6             | 5             |
| Risk Level             | -        | Low           | Medium        |

### AI Recommendation

**Alternative A — Recommended**  
Better balanced PPA with lower functional risk.

- 8% area reduction
- 4% power reduction
- 2% performance improvement
- Lower design risk
- Cleaner and more maintainable code
- Functionally equivalent (verified)
- Good balance for production use

#### Alternative B

- Higher optimization (15% area, 11% power)
- Slight performance loss (3%)
- Medium risk (more aggressive changes)
- Use for low-power or area-constrained designs



**Turn Old RTL into Living Knowledge**  
Find design intent. Understand history.  
Make confident changes.

### Fragmented Legacy Engineering Data

- Old RTL Code**  
\*.v \*.sv
- Old Specifications**  
PDF / DOC / XLS
- Old Bugs / Issues**  
Bug reports, CRs
- Old Testbenches**  
Tests, Scripts, Logs
- Old Engineers**  
Emails, Reviews, Comments
- Old Documentation**  
Design docs, meeting notes

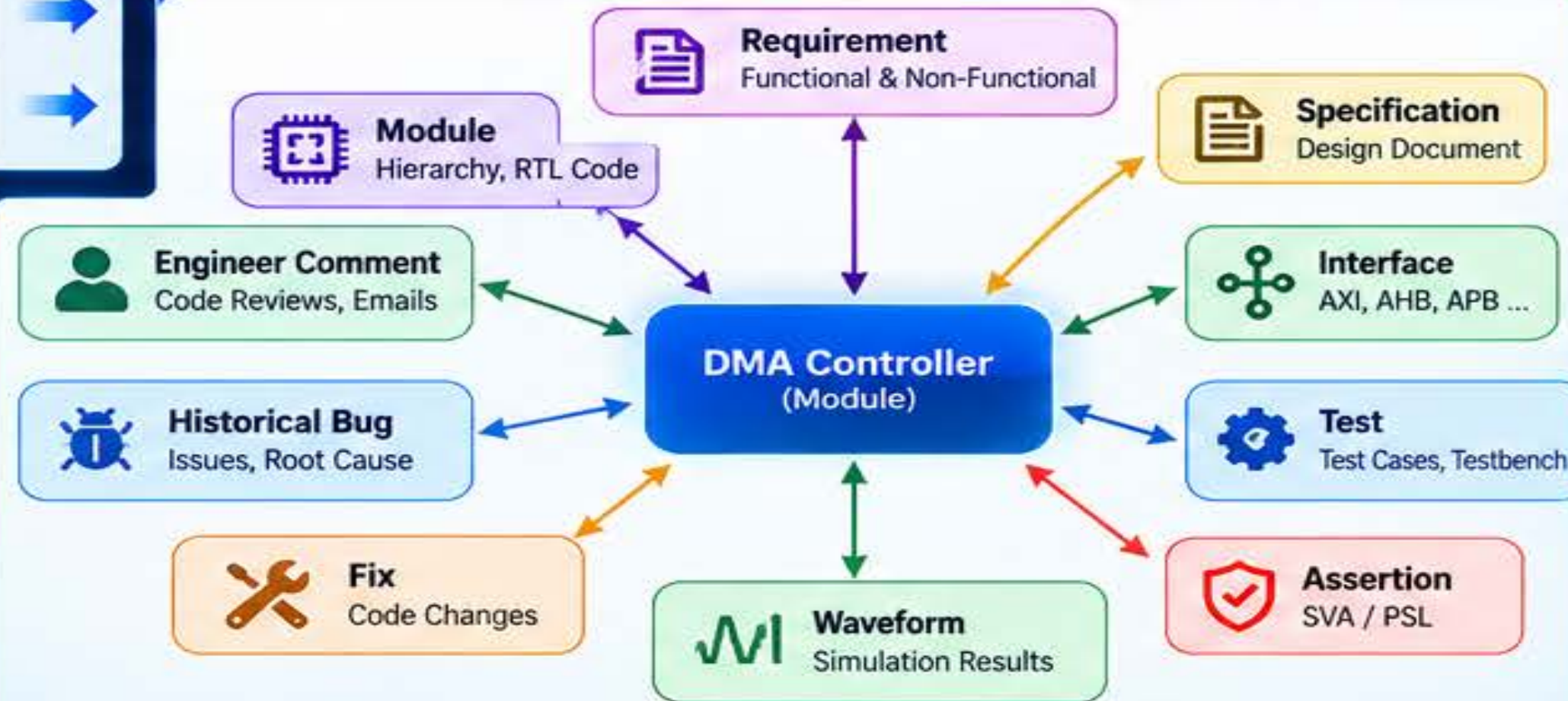
### SiliconAI — Legacy RTL Archaeology Engine



### Business Value

- ✓ Understand legacy IP faster
- ✓ Recover lost design knowledge
- ✓ Reduce engineer ramp-up time
- ✓ Enable safe modifications
- ✓ Preserve institutional knowledge
- ✓ Accelerate IP acquisition & maintenance
- ✓ Lower verification and debug time
- ✓ Turn old code into living documentation

### RTL Knowledge Graph (Example: DMA Controller)



### Ask SiliconAI (Natural Language)

Why was this unusual logic added in the DMA controller?

Ask

### AI Answer (with Source Links)

This logic was added to handle a FIFO overflow issue observed in high-burst scenarios on AXI interface.

### Design Intent (Summary)

- During high burst transfers, the write data rate could exceed the FIFO consumption rate.
- This logic throttles the AXI read requests when FIFO is near full.
- It was added as part of CR-482 to fix intermittent data loss observed in silicon.
- The design was reviewed and approved on 12 Aug 2018.
- Corresponding test case and assertion were also added.

### Source References

- RTL Code**  
dma\_ctrl.v (lines 234-248)
- Bug Report**  
CR-482: FIFO overflow (Jul 2018)
- Design Document**  
dma\_spec\_v2.1.pdf (section 4.3)
- Engineer Comment**  
"Added flow control to prevent overflow" (code review, Aug 2018)
- Test Case**  
test\_fifo\_overflow.sv
- Waveform**  
sim\_results/overflow.vcd

### RTL Code (Excerpt)

```

230 // Flow control to prevent FIFO overflow (CR-482)
231 if (fifo_level > FIFO_HIGH_TH) begin
232 axi_ar_valid <= 1'b0; // throttle read
233 end else if (read_req) begin
234 axi_ar_valid <= 1'b1;
235 end
236
237 // Added by SS - Aug 12, 2018
238 // During high burst AXI transfers
239 // to avoid FIFO overflow
240 // CR-482
241 /

```

### Design Evolution Timeline

- 2014** Initial RTL version (v1.0) – Basic DMA functionality
- 2016** Added AXI4 support (v1.2) – CR-210
- 2018** **FIFO flow control logic added** – CR-482  
Reason: Prevent FIFO overflow in high burst (See bug report)
- 2019** Added error handling – CR-510
- 2020** Added low power mode – CR-612
- 2022** Optimized for performance – CR-701
- 2024** Minor refactoring – CR-845

### Related Artifacts

- Requirement**: REQ-127: Support high burst transfer
- Specification**: dma\_spec\_v2.1.pdf (Section 4.3)
- Bug**: CR-482: FIFO overflow in high burst
- Fix Commit**: 1a2b3c4 - Add flow control logic
- Engineer**: S. Kumar (Added on Aug 12, 2018)
- Test Case**: test\_fifo\_overflow.sv
- Assertion**: axi\_fifo\_never\_overflow.sv
- Waveform**: sim\_results/overflow.vcd

### Impact Analysis

If I modify this logic, what could break?

### AI Impact Analysis

- 12** Modules affected
- 47** Signals affected
- 8** Assertions affected
- 23** Test cases affected
- 3** Interfaces affected
- 2** Coverage groups affected



### Multiple Design Artifacts as Input

- Requirements**  
PRD, Specs, User Stories
- Architecture**  
Block Diagrams, Docs
- RTL Code**  
Verilog / SystemVerilog
- Testbenches**  
Tests, Sequences
- Assertions**  
SVA / PSL
- Waveforms**  
Simulation Results
- Coverage**  
Reports, Databases
- Bug History**  
CRs, Issues, Fixes
- Documentation**  
Design Docs, Emails, Comments

### Knowledge Graph Explorer

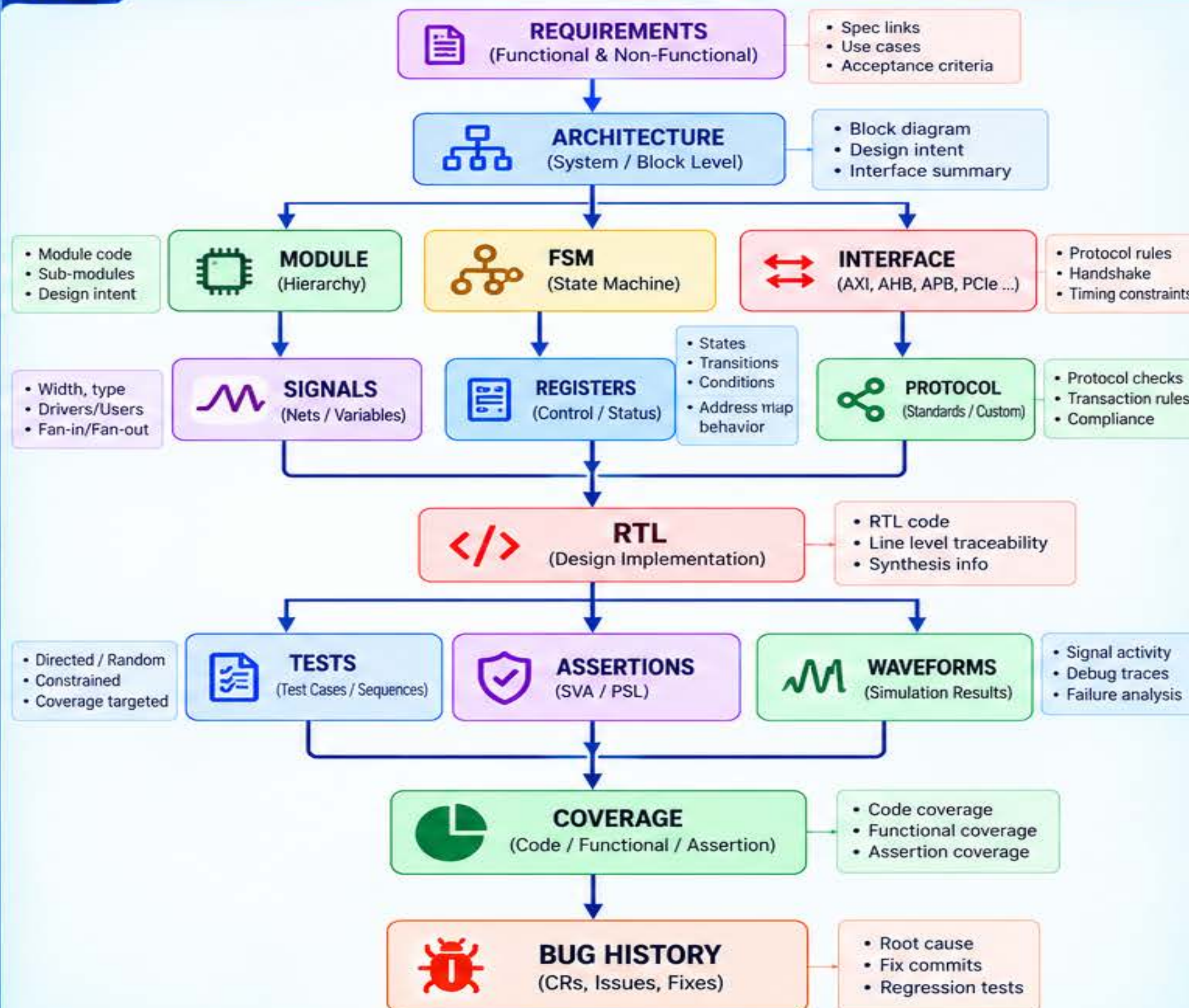
Search modules, signals, bugs, requirements...

- ▼ soc\_top
  - ▼ cpu\_subsystem
    - ▼ cpu\_core
      - alu
      - fsm\_ctrl
      - reg\_file
        - signals (32)
        - registers (64)
        - assertions (12)
        - tests (28)
        - coverage (85%)
        - bugs (3)
        - docs (5)
  - axi\_interface
  - mem\_ctrl
  - peripherals

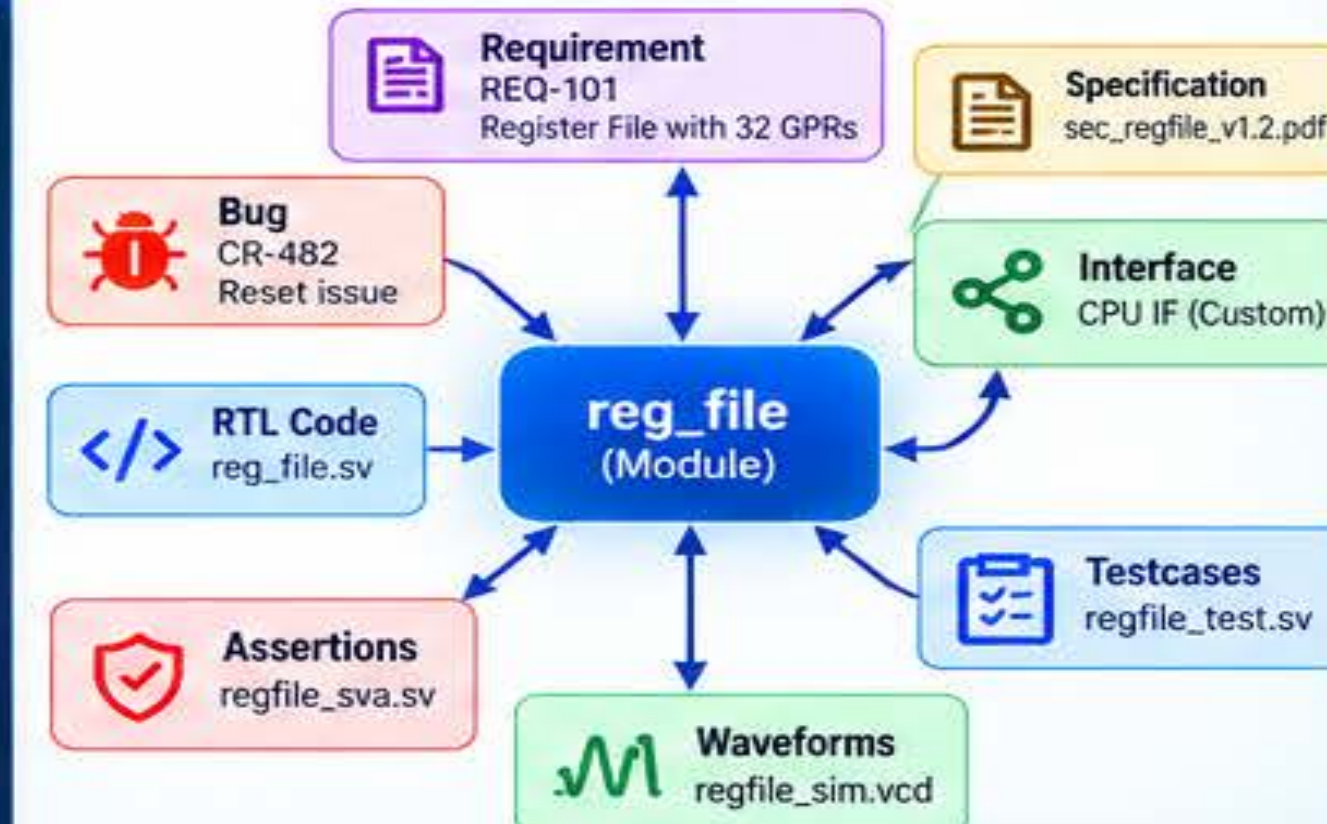


### SiliconAI Engineering Knowledge Graph (Example)

Entity → Relationship



### Example: Module Knowledge Graph



### Ask SiliconAI (Natural Language Q&A)

Why was this unusual logic added in reg\_file?

Ask

This logic was added to handle a reset race condition observed in CR-482. During power-on, the register file could output undefined values for 1 cycle. The fix (commit a1b2c3) added an extra reset synchronizer. See waveform and test case links below.

RTL (line 124)

CR-482 (bug)

Test case

Waveform

### Benefits

- ✓ Complete traceability from requirement to coverage
- ✓ Find design intent, history and context instantly
- ✓ Understand complex legacy RTL
- ✓ Faster debug, better design changes
- ✓ Preserve institutional knowledge
- ✓ Reduce engineer ramp-up time
- ✓ Enable AI-powered analysis, verification and refactoring



Turn Scattered  
Design Data into  
Unified Intelligence



1

## RTL Intelligence Score™

Every RTL block gets a comprehensive health score

### RTL HEALTH SCORE



|                 |             |    |
|-----------------|-------------|----|
| Code Quality    | <div></div> | 91 |
| Functional Risk | <div></div> | 78 |
| Verification    | <div></div> | 84 |
| Coverage        | <div></div> | 76 |
| Interface       | <div></div> | 89 |
| Maintainability | <div></div> | 81 |
| PPA             | <div></div> | 74 |
| Spec Compliance | <div></div> | 93 |

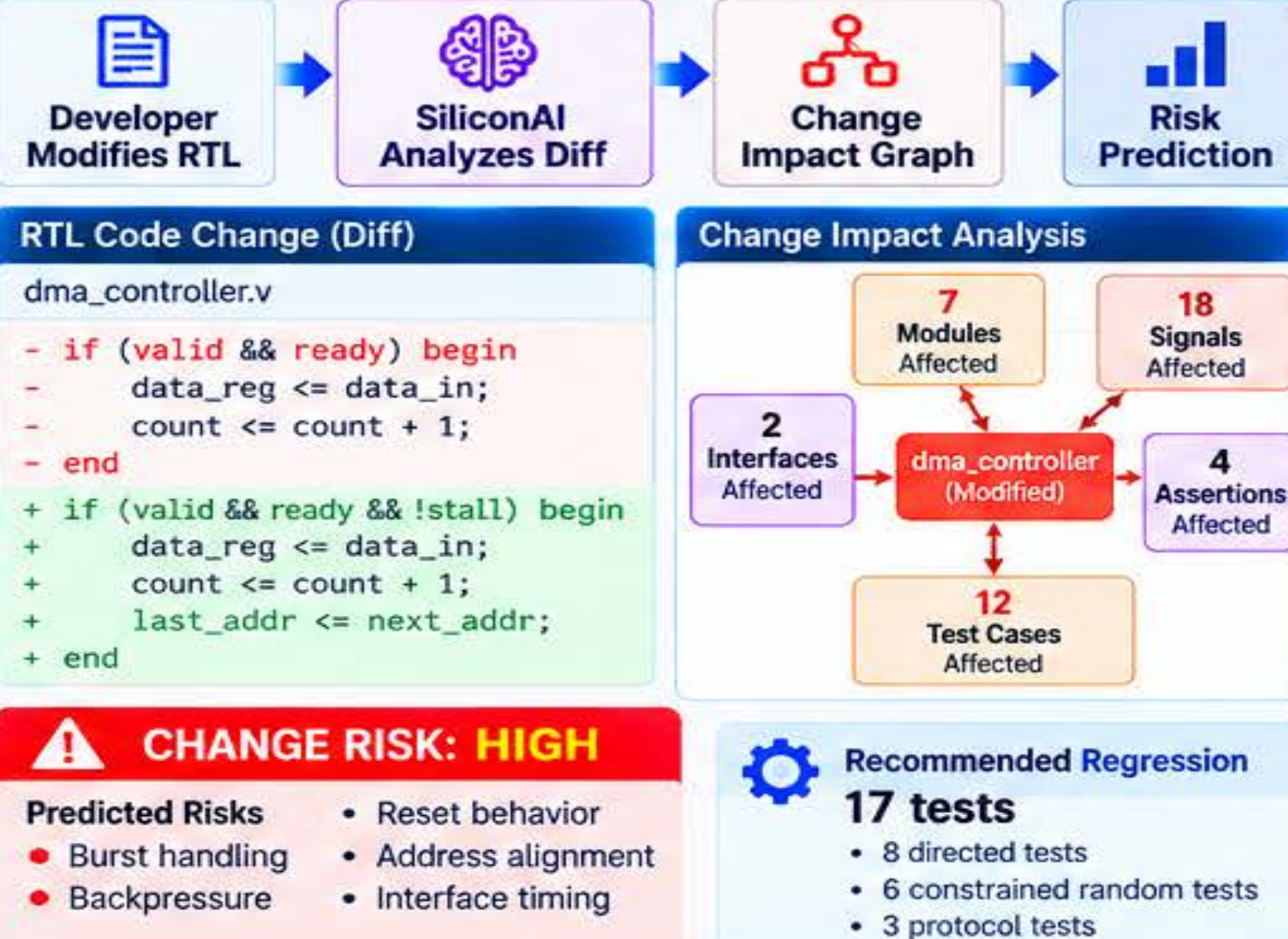
### Top RTL Blocks by Intelligence Score

| Rank | Module           | Score | Risk Level |
|------|------------------|-------|------------|
| 1    | axi_interconnect | 92    | Low        |
| 2    | cpu_core         | 88    | Low        |
| 3    | uart             | 85    | Low        |
| 4    | dma_controller   | 82    | Medium     |
| 5    | mem_ctrl         | 79    | Medium     |

2

## RTL Change Risk Prediction

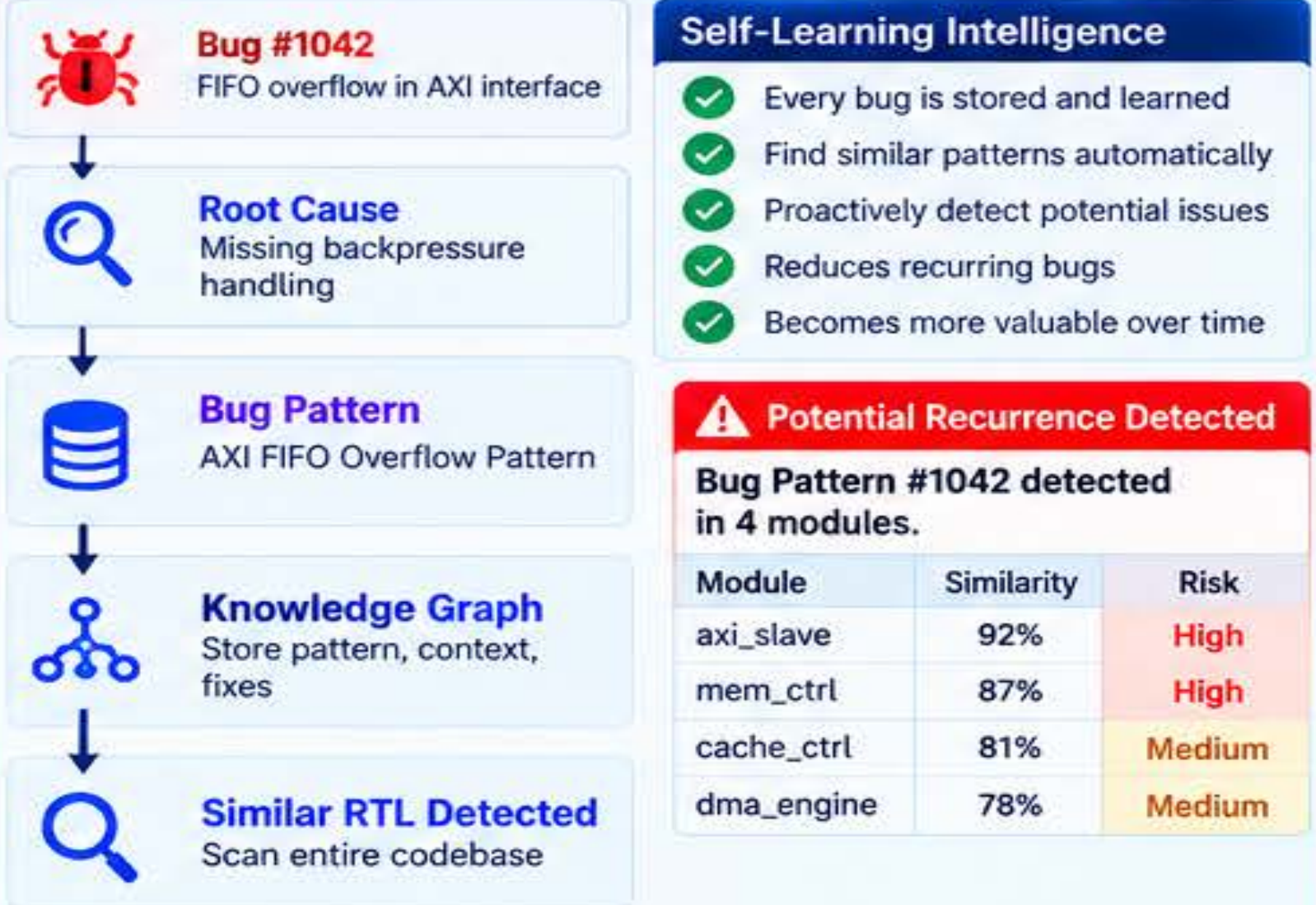
Predict impact before you commit



3

## Bug Pattern Intelligence

Turn every bug into organizational knowledge



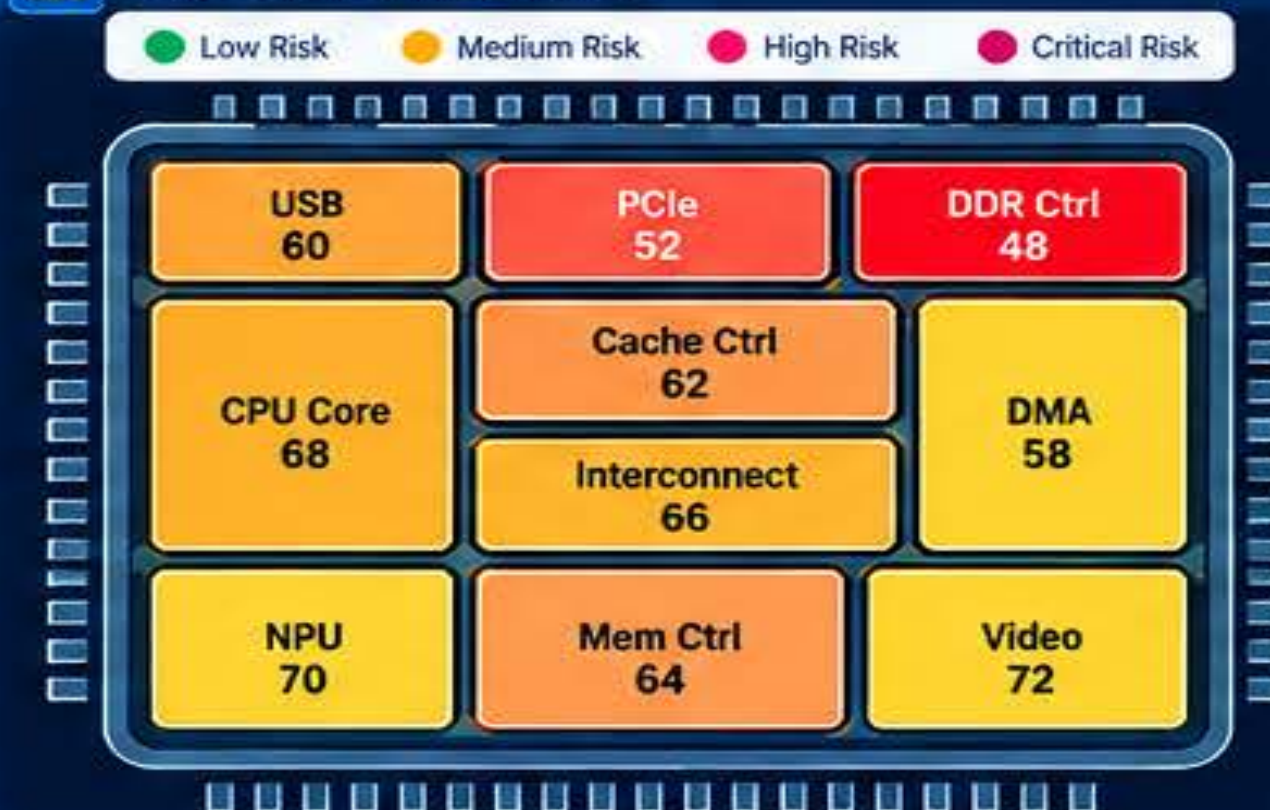
## RTL Risk Dashboard (SoC View)

Which 20 RTL blocks are highest risk across the entire SoC?

| Rank | Module           | Intelligence Score | Risk Level | Key Issues               |
|------|------------------|--------------------|------------|--------------------------|
| 1    | ddr_ctrl         | 48                 | Critical   | Reset, Timing, Coverage  |
| 2    | pci_e            | 52                 | Critical   | Protocol, Interface      |
| 3    | dma_controller   | 58                 | High       | Burst, Backpressure      |
| 4    | usb_ctrl         | 60                 | High       | CDC, Reset               |
| 5    | cache_ctrl       | 62                 | High       | Coherency, Coverage      |
| 6    | mem_ctrl         | 64                 | High       | Interface, Verification  |
| 7    | axi_interconnect | 66                 | High       | Ordering, Protocol       |
| 8    | cpu_core         | 68                 | Medium     | Coverage, PPA            |
| 9    | npu_top          | 70                 | Medium     | Verification, Complexity |
| 10   | video_encoder    | 72                 | Medium     | Timing, Maintainability  |

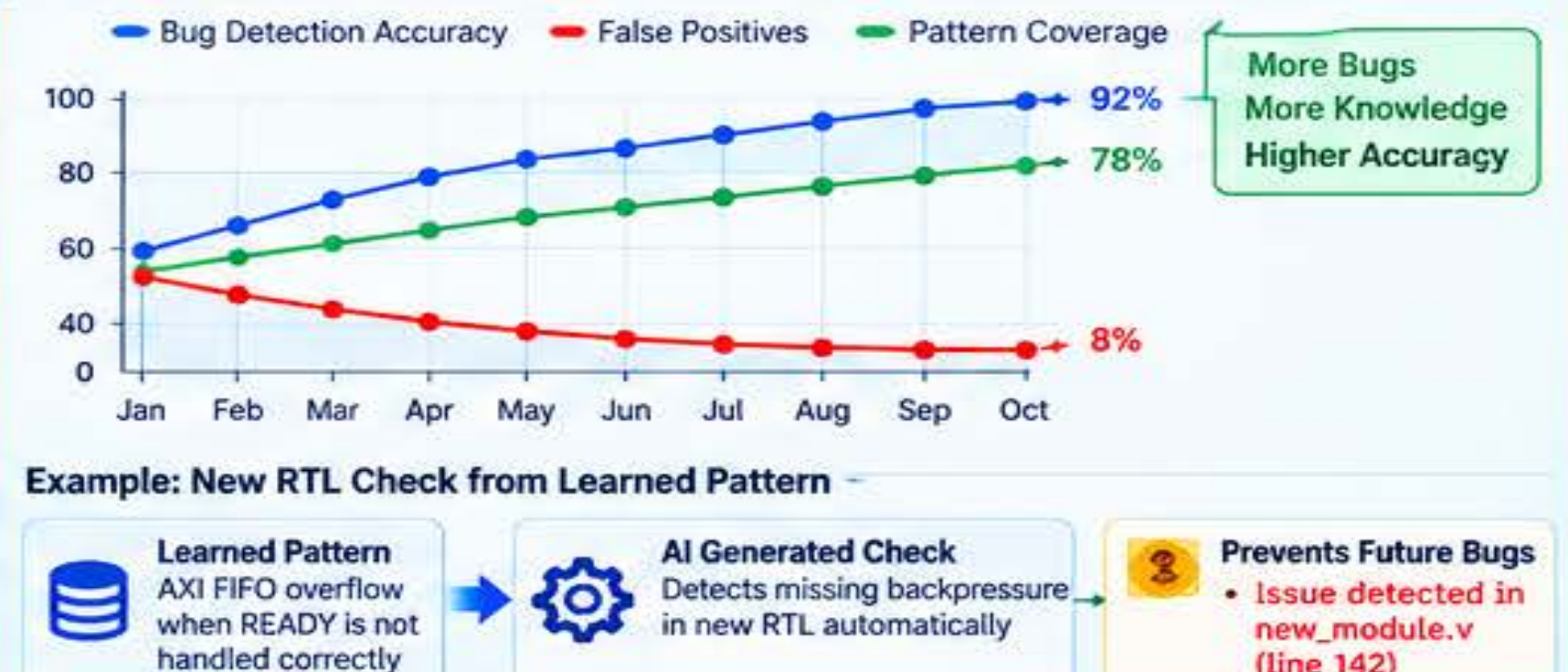


## SoC RTL Heatmap



## Continuous Learning from Bugs

The system gets smarter over time





## INPUTS (Multi-Source Engineering Data)

- RTL Code**  
(Verilog / SystemVerilog)
- Specifications**  
(PRD, Functional Spec)
- Architecture**  
(Block Diagrams)
- Testbenches**  
(Tests, Sequences)
- Simulation Logs**  
(Waveforms, Traces)
- Lint Reports**  
(Static Analysis)
- Bug Database**  
(CRs, Issues, Fixes)
- Documentation**  
(Design Docs, Emails)
- Historical Data**  
(Past Projects)

## SiliconAI RTL INTELLIGENCE

### 1 RTL ANALYSIS (Understand the RTL)

- Code Analysis**  
(Style, Quality, Complexity)
- Lint Intelligence**  
(Issues, Waivers, Best Practices)
- FSM Analysis**  
(State Machine Extraction)
- Dependency AI**  
(Design Hierarchy & Signal Flow)
- Interface Intelligence**  
(AXI, AHB, APB, PCIe, Custom)

### 2 DEBUG INTELLIGENCE (Find and Fix Issues)

- Log Analysis**  
(Simulation Logs, Errors)
- Waveform AI**  
(Intelligent Waveform Debug)
- Failure Analysis**  
(Test Failures, Assertions)
- Root Cause Analysis**  
(Explain Why & How to Fix)
- Debug AI**  
(Interactive Debug Assistant)

### 3 SPEC INTELLIGENCE (Connect Spec ↔ RTL)

- Spec Analysis**  
(Extract Requirements)
- RTL → Spec (Reverse Engineering)**  
(Generate Specification)
- Spec ↔ RTL Traceability**  
(Requirements, Coverage)
- Compliance Analysis**  
(Standards, Protocols, Guidelines)
- Legacy RTL Archaeology**  
(Reconstruct Design History)

## RTL KNOWLEDGE GRAPH (Central Engineering Intelligence Layer)

- |            |              |                    |
|------------|--------------|--------------------|
| Modules    | Requirements | Bugs & Fixes       |
| Signals    | Test Cases   | Design History     |
| Registers  | Assertions   | Engineer Knowledge |
| FSMs       | Waveforms    | Specifications     |
| Interfaces | Coverage     | Dependencies       |

- AI-POWERED QUERY**
- "Why was this logic added?"
  - "What can break if I change this?"
  - "Show me all related tests"
  - "Find similar bug patterns"
  - "Explain this waveform"

### 4 VERIFICATION INTELLIGENCE (Ensure Correctness)

- Coverage AI**  
(Code, Functional, Assertion)
- Assertion AI**  
(SVA Generation & Analysis)
- Test Generation**  
(Directed, Random, Constrained)
- Regression AI**  
(Intelligent Test Selection)

### 5 OPTIMIZATION INTELLIGENCE (Better PPA & Quality)

- AI Refactoring**  
(Cleaner, Safer RTL)
- PPA Analysis**  
(Area, Power, Performance)
- Generate Alternatives**  
(Multiple Implementations)
- Equivalence Checking**  
(Formal / Simulation)

### 6 PREDICTION INTELLIGENCE (Prevent Issues)

- Change Risk Prediction**  
(Pre-commit Analysis)
- Bug Pattern Intelligence**  
(Learn from Past Bugs)
- Impact Analysis**  
(Modules, Signals, Tests, Interfaces)
- Failure Prediction**  
(High-risk Areas, Hotspots)

## OUTPUTS (Actionable Insights)

- RTL Health Score**  
(Risk, Quality, Compliance)
- Top Issues & Recommendations**
- Interactive Debug Reports**
- Generated Specification**
- Traceability Matrix**
- Verification Plan**
- Optimization Recommendations**
- Change Impact Analysis**
- Bug Predictions & Prevention**
- AI Chat Assistant**

## BUSINESS IMPACT

- Higher RTL Quality
- Faster Debug & Verification
- Reduced Engineering Effort
- Lower Design Risk
- Better PPA and Performance
- Preserve Engineering Knowledge
- Faster Engineer Onboarding
- Accelerate Time-to-Silicon

**Iterative Improvement**  
(Self-Learning System)



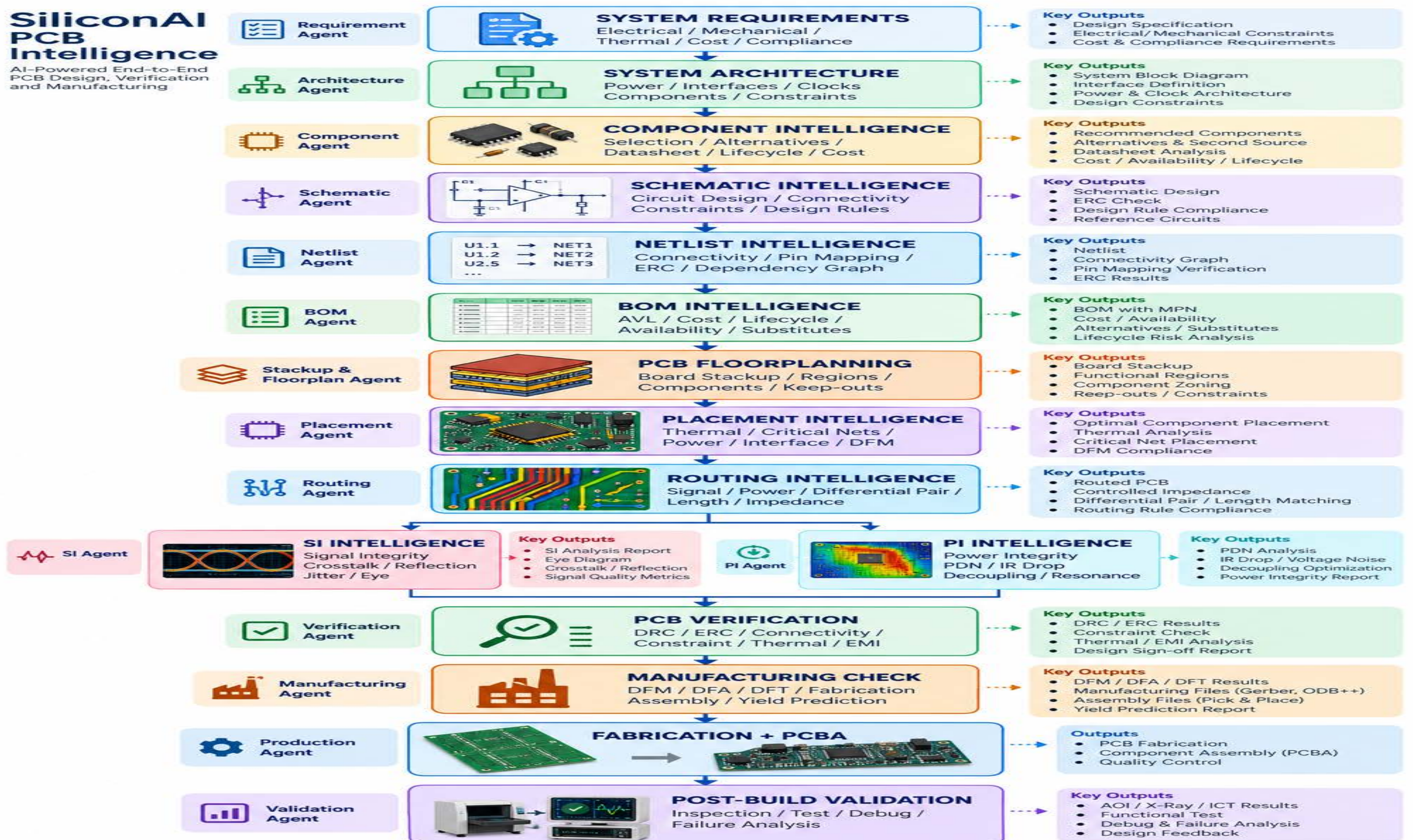
**VERIFIED RTL**

Higher Quality | Lower Risk | Faster Time-to-Market



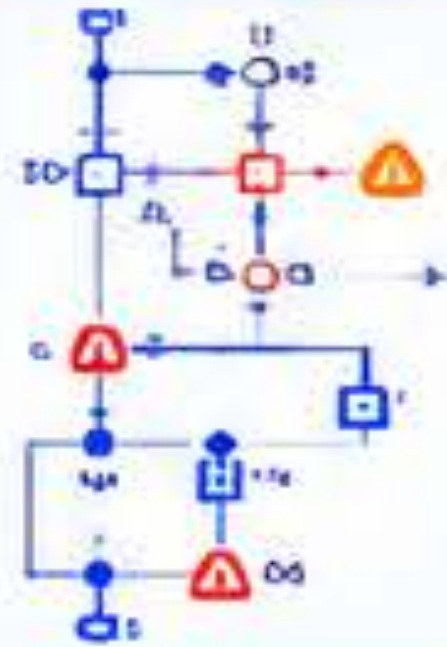
**Continuous Learning**  
(Every Project Makes It Smarter)

# **PCB - Design and Verification**



# PCB VERIFICATION

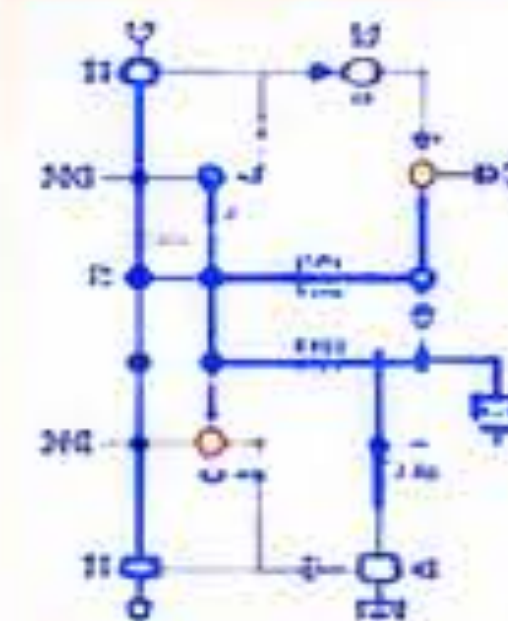
AI-Powered End-to-End PCB Verification Flow



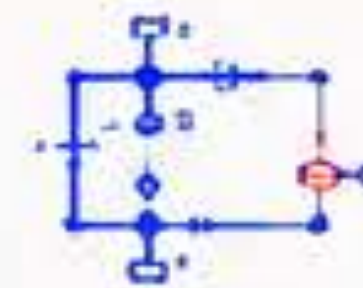
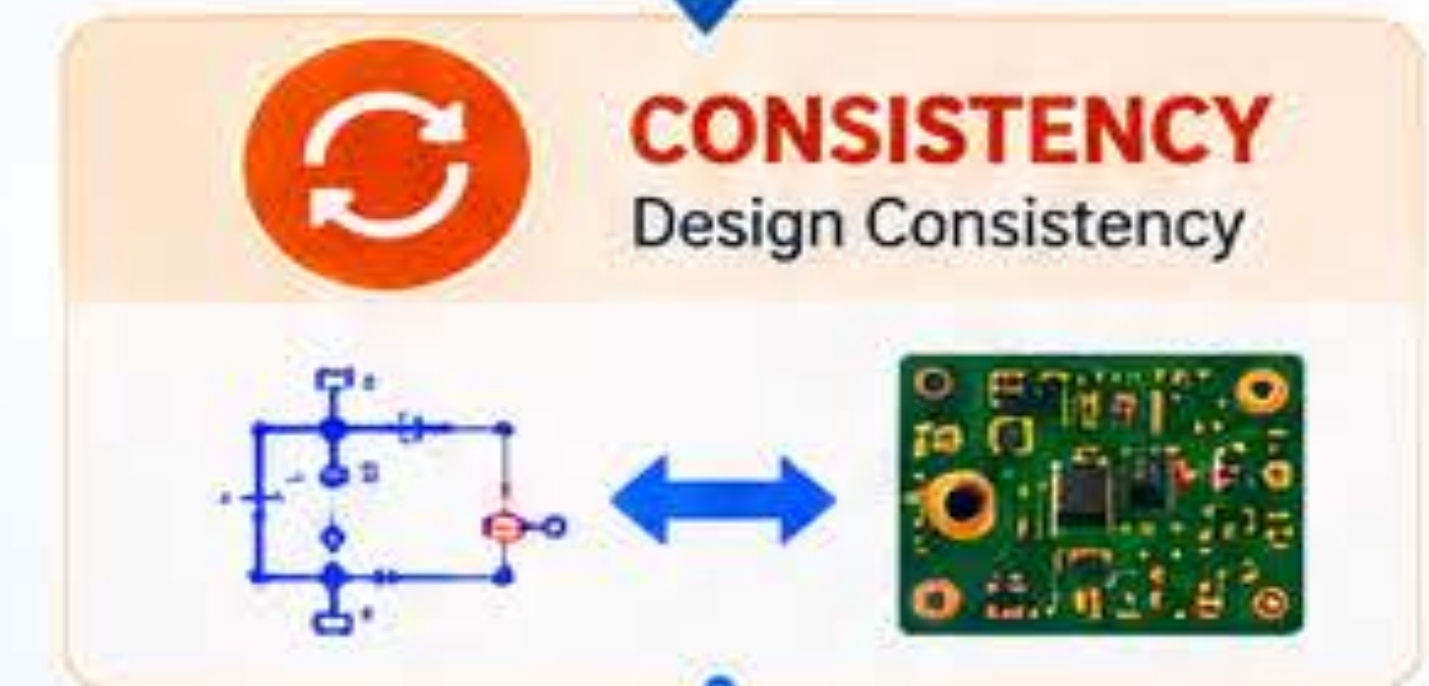
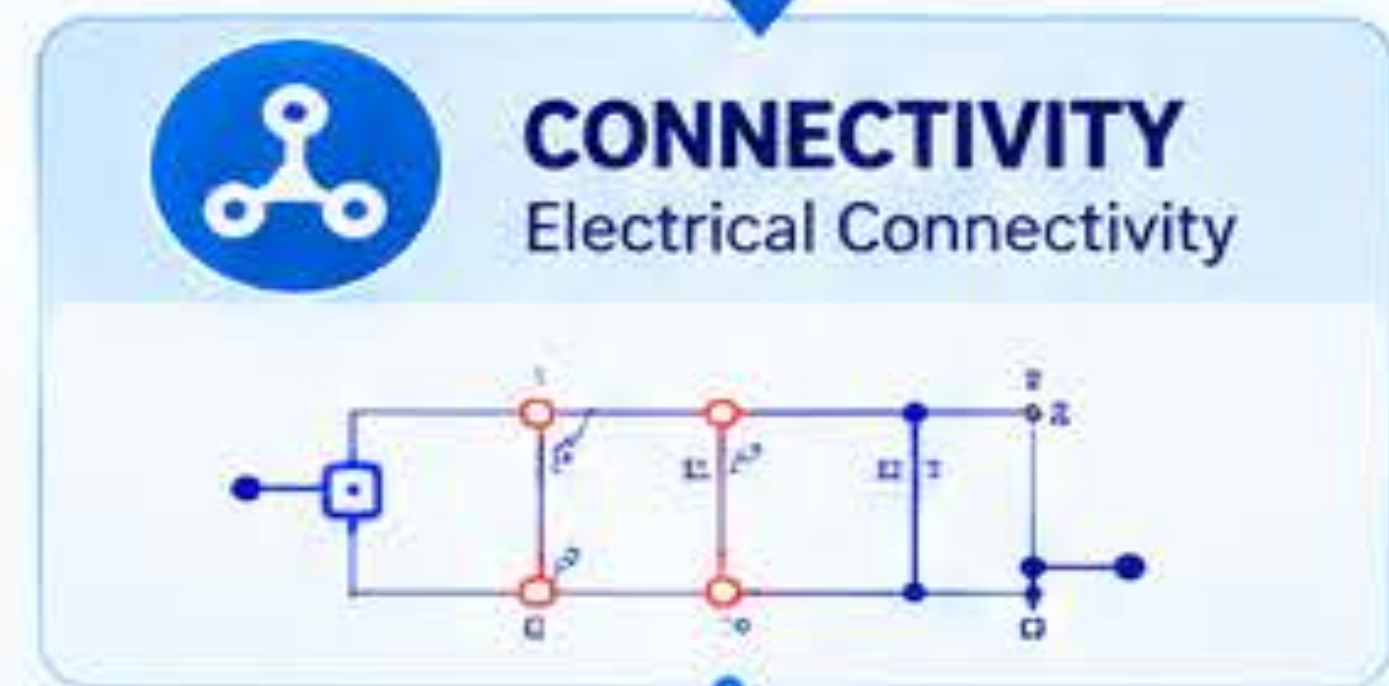
- ✓ Power pin checks
- ✓ Floating pins
- ✓ Input/Output conflicts
- ✓ Missing power
- ✓ Wrong voltage
- ✓ Unconnected pins

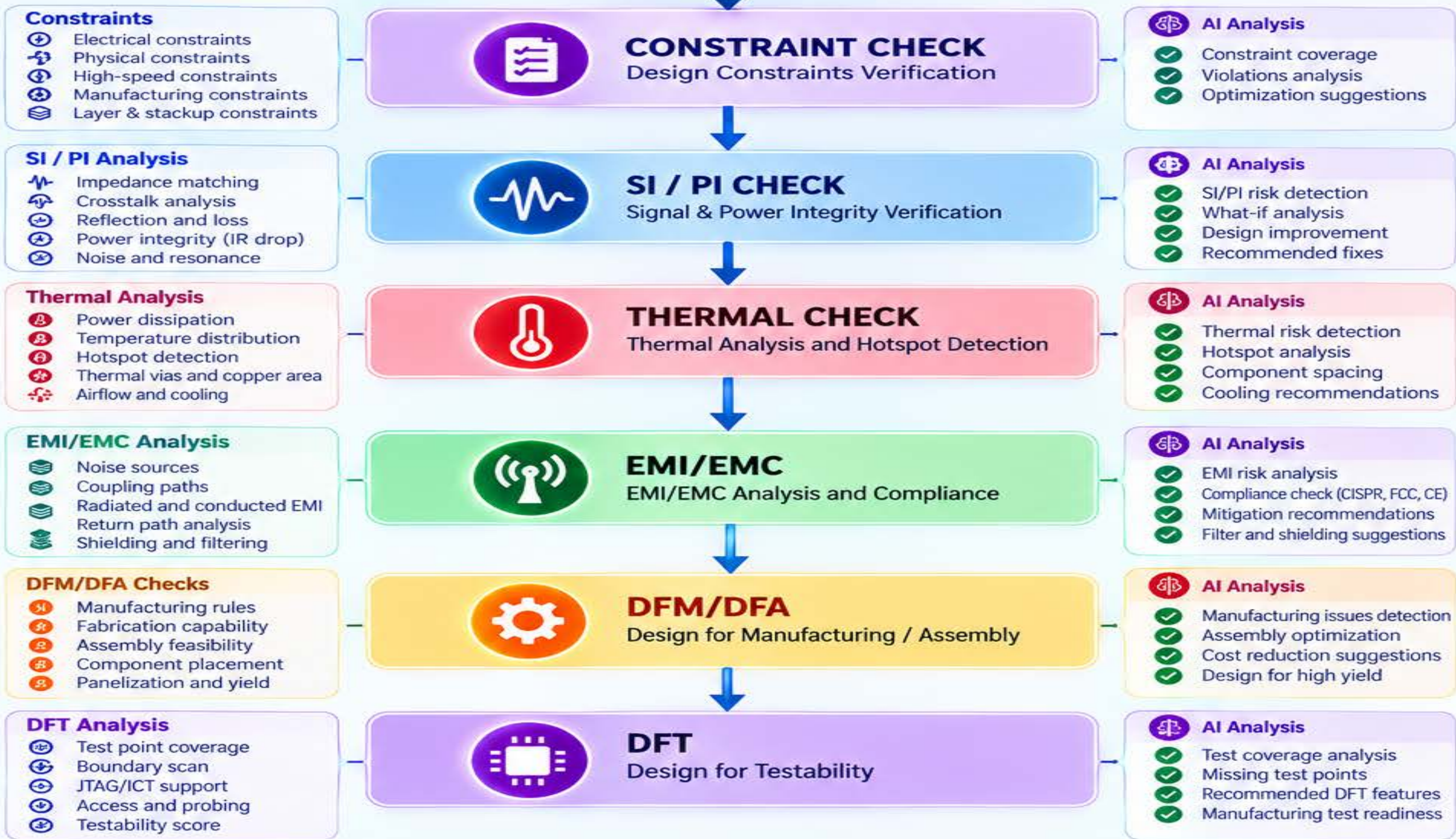


- ✓ Clearance
- ✓ Trace width
- ✓ Spacing
- ✓ Via / Hole
- ✓ Layer rules
- ✓ Solder mask



- ✓ Schematic ↔ PCB
- ✓ Connectivity check
- ✓ Missing / extra nets
- ✓ Component mapping
- ✓ Pin mapping
- ✓ Consistency check





## 1 Requirements Intelligence PCB Requirement Agent

Understand the product and system requirements before schematic design.

### Input Documents & Requirements

- System Specification
- Product Requirements
- Electrical Requirements
- Mechanical Requirements
- Thermal Requirements
- Cost Targets
- Safety Requirements
- Regulatory Requirements
- Interface Requirements
- Manufacturing Constraints

### AI Analysis

- Voltage / current requirements
- Interfaces (USB, PCIe, Ethernet...)
- Clock requirements
- Memory requirements
- Processor / MCU / FPGA needs
- Environmental requirements
- Temperature range
- Board dimensions
- Layer constraints
- Cost constraints
- Manufacturing constraints

### Output – Design Constraints

- Electrical Constraints**  
Voltage, Current, Power, Interfaces
- Mechanical Constraints**  
Board size, Form factor, Mounting
- Thermal Constraints**  
Temperature range, Heat dissipation
- Manufacturing Constraints**  
Layer count, Technology, DFM rules
- Compliance Constraints**  
Safety, Regulatory, EMI/EMC

### PCB Design Specification

- System Requirements Summary
- Design Constraints
- Interface List
- Power / Clock / Memory Requirements
- Mechanical Specification
- Thermal Targets
- Cost Targets
- Regulatory Compliance
- Manufacturing Guidelines

## 2 System Architecture Intelligence Architecture Agent

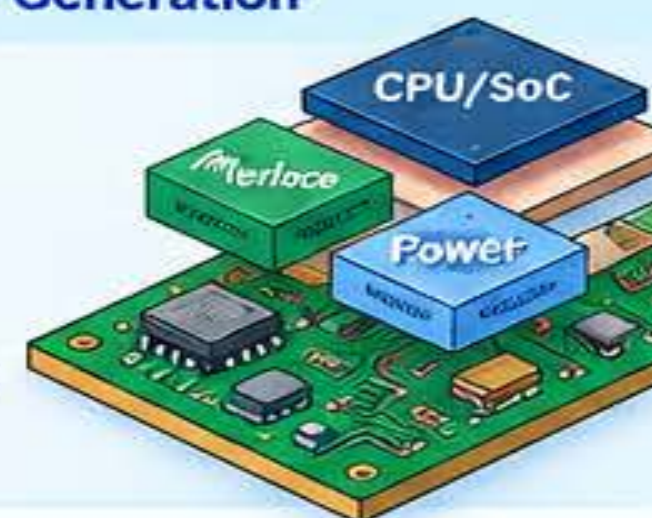
Convert requirements to hardware architecture and identify subsystems, interfaces and key design elements.

### Input: Requirements & Constraints

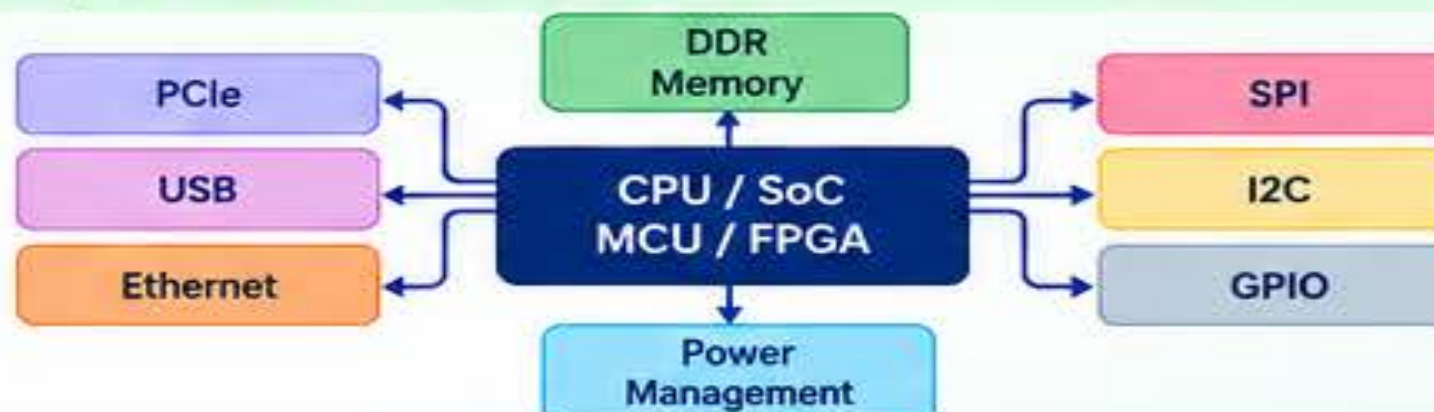
- PCB Design Specification
- Functional Requirements
- Performance Targets
- Cost & Compliance

### AI Analysis & Architecture Generation

- Functional decomposition
- Identify subsystems
- Determine required components
- Define interfaces and protocols
- Power domain planning
- Clock domain planning
- Identify high-speed and critical nets
- Thermal hotspot analysis
- Design constraints and trade-offs



### Output: Hardware Architecture



### Architecture Outputs

- System Block Diagram
- Subsystem Definition
- Interface Definition
- Power & Clock Architecture
- Component Requirements
- High-Speed / Critical Nets
- Thermal Hotspots
- Design Constraints

## 3 Component Intelligence Component Agent

Select the right components using AI analysis of datasheets, electrical compatibility, cost, availability and lifecycle.

### Input: Component Requirements

- Required Functions
- Electrical Specifications
- Interface Requirements
- Performance Targets
- Package Constraints
- Cost Targets
- Lifecycle / Availability

### AI Component Analysis

- Search candidate components
- Analyze datasheets (latest revisions)
- Evaluate electrical specifications
- Check package and pinout
- Verify operating voltage/current
- Check temperature range
- Evaluate availability and lead time
- Compare cost and manufacturers
- Check lifecycle (active / NRND / EOL)
- Find second-source alternatives

### Component Selection Flow



### Component Substitution



### Outputs

- Recommended Components (with confidence score)
- Alternative Components (second source)
- Detailed Datasheet Summary
- Electrical Compatibility Report
- Pinout and Package Information
- Cost and Availability Analysis
- Lifecycle Risk Analysis
- BOM Suggestions



## Schematic Intelligence

AI-assists in schematic creation and analysis



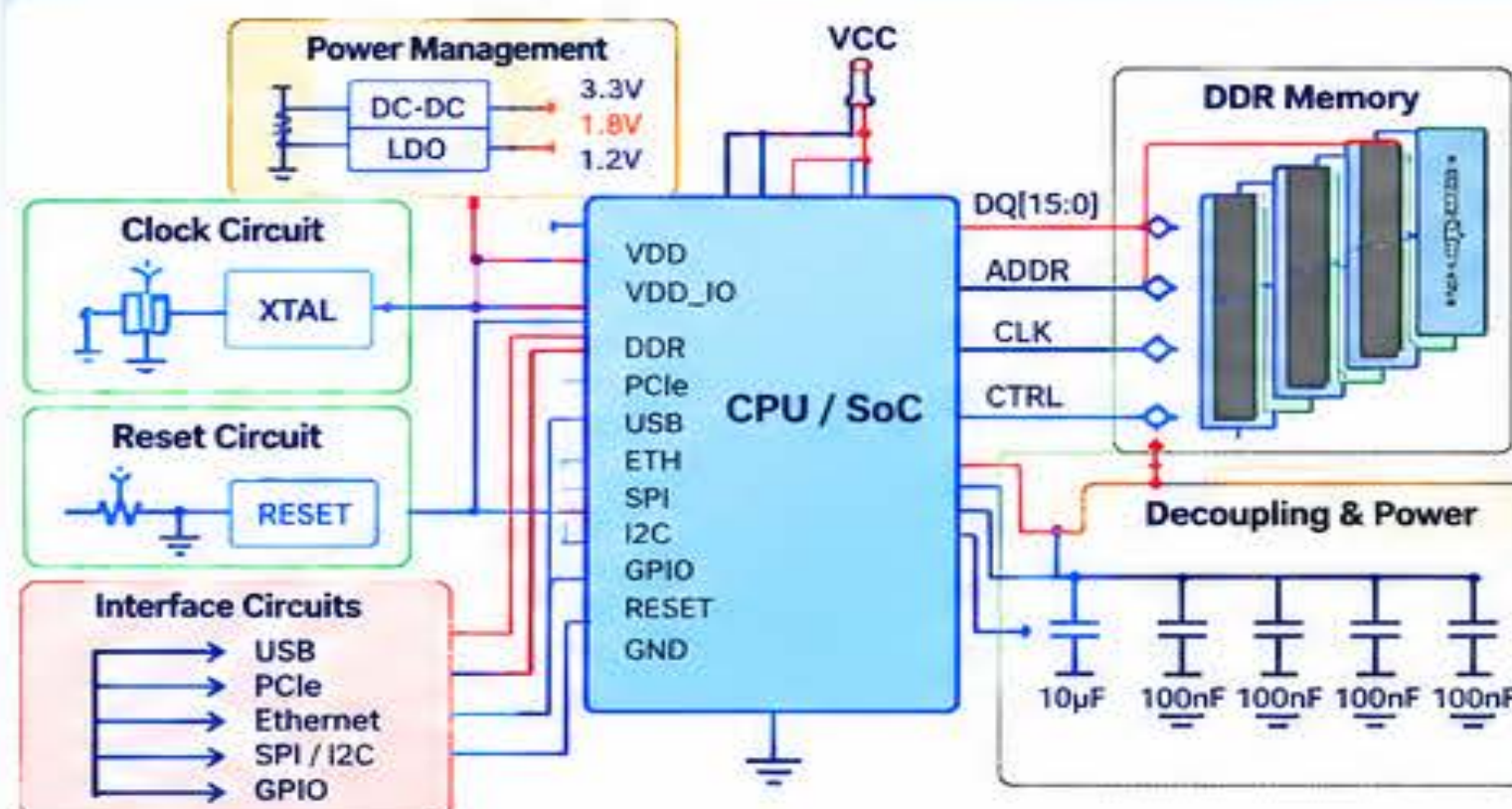
Schematic Agent



### AI-Assisted Schematic Creation

- ⚙️ Use architecture and component selection
- 🔧 Automatically create schematic blocks
- 💡 Suggest reference circuits
- 📄 Apply design rules and best practices
- 🔗 Generate complete schematic
- 🔄 Iterative refinement with engineer feedback

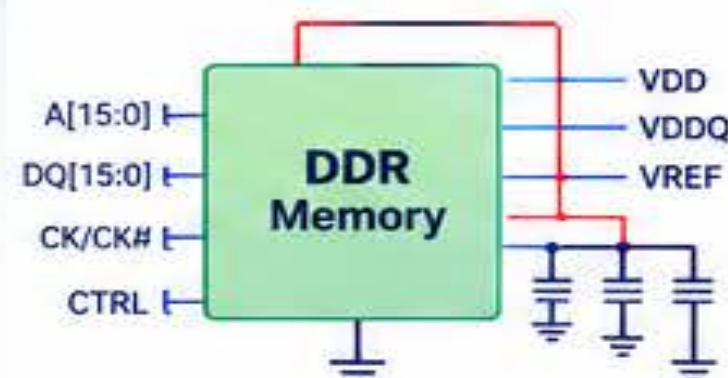
Example: CPU + DDR + Power Schematic (Generated with AI Assistance)



### AI Schematic Analysis & Verification

- ⚠️ Missing connections
- ⚠️ Incorrect connections
- ⚠️ Floating pins
- ⚡ Power pins
- 🔌 Pull-up / Pull-down requirements
- 🔌 Decoupling capacitors
- 🔌 Termination resistors
- 🔌 Protection circuits
- 🕒 Clock circuits
- 🔄 Reset circuits
- 🔗 Interface circuits
- ⚙️ Design rule compliance

Example: DDR Interface Analysis



- ✅ Power (VDD, VDDQ, VPP)
- ✅ Ground (multiple GND pins)
- ✅ Address lines (A[15:0])
- ✅ Data lines (DQ[15:0])
- ✅ Differential clocks (CK/CK#)
- ✅ Termination resistors (ODT)
- ✅ Decoupling capacitors
- ✅ Reference voltage (VREF)
- ✅ Required topology (length match)

### AI Recommendations

#### Issues Found

- ⚠️ Decoupling capacitor missing on VDDQ
- ⚠️ Unconnected RESET pin
- ⚠️ Pull-up resistor required on I2C
- ⚠️ Differential pair length mismatch
- ⚠️ Unused GPIO pins floating

#### AI Suggestion

- 🔧 Add 100nF + 1µF decoupling
- 🔧 Connect to reset circuit with RC
- 🔧 Add 4.7kΩ pull-up resistors
- 🔧 Match length within 5 mils
- 🔧 Tie unused GPIO to GND or VCC

#### Auto-Fix / Apply

- ✅ Components automatically added
- ✅ Update schematic
- ✅ Verify design rules
- ✅ Re-analyze
- ✅ Generate updated schematic



## Netlist Intelligence

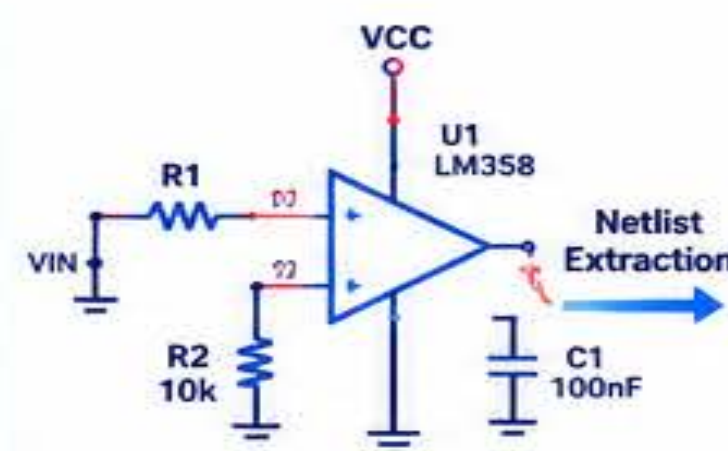
AI-powered connectivity, verification and consistency



Netlist Agent



### From Schematic to Netlist



### Generated Netlist (Example)

```
* LM358 Amplifier
U1 LM358 VCC VIN VOUT GND
R1 VIN N001 10k
R2 N001 GND 10k
C1 VCC GND 100nF
.END
```

### Connectivity Graph

(Component - Pin - Net)



### AI Netlist Analysis

- ⚠️ Missing nets
- ⚠️ Unexpected nets
- 🔗 Incorrect pin mapping
- 🔗 Short circuits
- 🔗 Open circuits
- ⚡ Power connectivity problems
- 🔌 Ground connectivity problems
- 🔌 Differential-pair mismatches
- 🔌 Net naming inconsistencies
- 🔄 Schematic ↔ PCB mismatch

### Example: Netlist Issues & AI Analysis

| Issue                         | Location       | AI Analysis                                     | AI Suggestion                     |
|-------------------------------|----------------|-------------------------------------------------|-----------------------------------|
| ⚠️ Unconnected Pin            | U3.5           | NC pin should be connected to GND or configured | Connect to GND (or per datasheet) |
| ⚠️ Missing Decoupling         | U1 (VDD)       | No decoupling cap on power pin                  | Add 100nF + 1µF capacitors        |
| ⚠️ Short Circuit              | Net: VCC       | VCC shorted to GND in netlist                   | Check schematic connection        |
| ⚠️ Differential Pair Mismatch | USB_DP/USB_DN  | Length mismatch > 100 mils                      | Match lengths within 5 mils       |
| ✅ Net Naming Inconsistency    | I2C_SDA vs SDA | Different naming in schematic and PCB           | Use consistent net names          |
| ⚡ Power Connectivity          | DDR_VREF       | VREF not connected to reference source          | Connect to VREF generator         |

### Schematic ↔ Netlist ↔ PCB Consistency



### Outputs

- ✅ Verified Netlist
- ✅ Connectivity Graph
- ✅ Pin Mapping Report
- ✅ Net Consistency Report
- ✅ Schematic ↔ PCB Match Report
- ✅ Issues & Recommendations
- ✅ Updated Schematic / Netlist
- ✅ Design Rule Check Report



Reduce Cost



Ensure Availability



Mitigate Obsolescence



Right Alternatives



## Schematic

Circuit Design with Components



## BOM

Component List (Extract from Schematic)



## AI Analysis

Evaluate, Compare and Recommend



## Optimized BOM

Cost Effective  
Reliable & Available Design



## BOM Intelligence Agent

AI-analyzes each component for technical, commercial and lifecycle parameters.



Extract BOM from Schematic (automatic parsing)



Evaluate component specifications



Check availability and suppliers



Compare cost and alternatives



Analyze lifecycle and obsolescence



Recommend best alternatives with compatibility score



Optimize BOM for cost, availability and reliability

## BOM Analysis Parameters



### Technical Analysis

- ✓ Manufacturer
- ✓ MPN (Manufacturer Part Number)
- ✓ Package (SMD / Through-hole)
- ✓ Electrical Rating (Voltage / Current)
- ✓ Tolerance
- ✓ Temperature Range
- ✓ Power Rating
- ✓ Speed / Frequency
- ✓ Pinout & Compatibility



### Commercial Analysis

- ✓ Unit Cost
- ✓ MOQ (Minimum Order Quantity)
- ✓ Lead Time
- ✓ Availability (Stock)
- ✓ Supplier Information
- ✓ Multiple Supplier Comparison
- ✓ Price Breaks (Volume Pricing)
- ✓ Total Cost Analysis



### Lifecycle Analysis

#### ACTIVE

Recommended for new designs

#### NRND

Not recommended for new designs

#### EOL

End of life

#### OBSOLETE

Replace immediately

## BOM Dashboard (Example)

### Component Status



|                 |    |
|-----------------|----|
| Active          | 85 |
| NRND            | 12 |
| EOL             | 8  |
| Obsolete        | 5  |
| Not Recommended | 10 |

### Cost Analysis

Total Cost  
₹ 12,450

Potential Savings  
₹ 3,850 (31%)

## BOM Analysis Example

| # | Component      | Value / Part      | Manufacturer       | Package  | Voltage / Current | Unit Cost (₹) | Availability        | Lifecycle | AI Recommendation       |
|---|----------------|-------------------|--------------------|----------|-------------------|---------------|---------------------|-----------|-------------------------|
| 1 | U1 - MCU       | STM32F407VGT6     | STMicroelectronics | LQFP-100 | 3.3V              | 650           | ✓ In Stock          | ✓ Active  | ✓ Keep                  |
| 2 | U2 - DDR3      | MT41K256M16HA-125 | Micron             | FBGA-96  | 1.5V              | 320           | ✓ Limited (4 weeks) | ✓ NRND    | ✓ Alternative Available |
| 3 | U3 - LDO       | TPS7A4700RGWR     | TI                 | VQFN-20  | 3.3V / 1A         | 120           | ✓ In Stock          | ✓ Active  | ✓ Keep                  |
| 4 | U4 - Ethernet  | LAN8720A          | Microchip          | QFN-24   | 3.3V              | 180           | ✗ Limited           | ✗ EOL     | ✗ Replace Recommended   |
| 5 | C1 - Capacitor | 10µF 25V          | Murata             | 0603     | 25V               | 2.5           | ✓ In Stock          | ✓ Active  | ✓ Keep                  |
| 6 | R1 - Resistor  | 10kΩ 1%           | Yageo              | 0603     | 0.1W              | 0.3           | ✓ In Stock          | ✓ Active  | ✓ Keep                  |

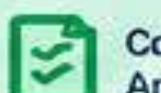
## Component Alternative Selection Flow



Original Component



Find Alternatives



Compatibility Analysis

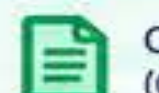


Cost & Availability



Score & Recommend

## Outputs



Optimized BOM (Cost & Availability)



Alternative Components



Lifecycle Risk Report



Supplier Comparison



Cost Savings Report

## Alternative Components for EOL/NRND Part

Example: Ethernet PHY (Original: LAN8720A - Microchip)

| Option   |  | Manufacturer | Part Number | Package | Unit Cost (₹) | Availability | Compatibility Score | Status      |
|----------|--|--------------|-------------|---------|---------------|--------------|---------------------|-------------|
| Original |  | Microchip    | LAN8720A    | QFN-24  | 180           | Limited      | 100%                | EOL         |
| Alt 1    |  | TI           | DP83848     | QFN-24  | 165           | In Stock     | 95%                 | Recommended |
| Alt 2    |  | Realtek      | RTL8201F    | QFN-24  | 120           | In Stock     | 90%                 | Good        |
| Alt 3    |  | Microchip    | LAN8742A    | QFN-24  | 150           | In Stock     | 92%                 | Good        |



## PCB Stackup Intelligence

AI-optimized stackup for performance, manufacturability and cost

Stackup  
Agent

## Input: Board Requirements

- High-speed interfaces (PCIe, DDR, USB, Ethernet...)
- Impedance requirements (Single-ended / Differential)
- Signal and power integrity targets
- Layer count constraints
- Board thickness / form factor
- Thermal requirements
- Manufacturing capability (fabrication rules)
- Cost targets



## AI Analysis &amp; Optimization

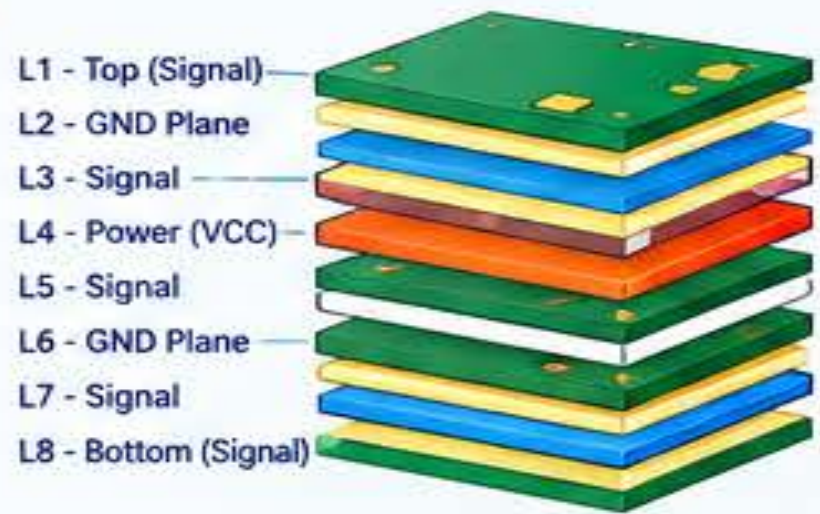
- Analyze signal, power and ground requirements
- Recommend optimal layer count
- Select dielectric materials (FR4, low-loss, etc.)
- Determine copper thickness
- Calculate controlled impedance (single-ended / differential)
- Optimize power/ground planes
- Minimize EMI and crosstalk
- Check manufacturability and cost
- Provide multiple stackup options with PPA analysis



## AI Considerations

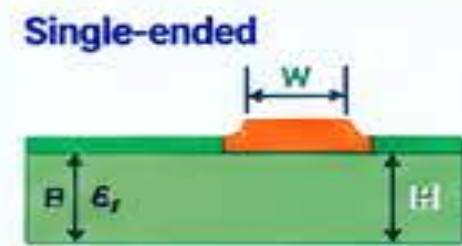
- Signal layers
- Power layers
- Ground planes
- Controlled impedance
- Differential pairs
- High-speed interfaces
- EMI / crosstalk
- Thermal performance
- Manufacturing capability
- Cost optimization

## Optimized Stackup Example (8-Layer)



| Layer | Type   | Material / Dielectric | Thickness (μm) | Description            |
|-------|--------|-----------------------|----------------|------------------------|
| L1    | Signal | Low-loss FR4          | 35             | High-speed signals     |
| L2    | Ground | FR4                   | 70             | Continuous GND plane   |
| L3    | Signal | Low-loss FR4          | 35             | High-speed / sensitive |
| L4    | Power  | FR4                   | 70             | VCC plane              |
| L5    | Signal | Low-loss FR4          | 35             | General signals        |
| L6    | Ground | FR4                   | 70             | Continuous GND plane   |
| L7    | Signal | Low-loss FR4          | 35             | Differential pairs     |
| L8    | Signal | FR4                   | 35             | General signals        |

## Impedance Calculation &amp; Validation



Target: 50Ω  
Calculated: 49.8Ω ✓

## Differential Pair



Target: 100Ω  
Calculated: 99.6Ω ✓

## Stackup Options Comparison

| Parameter          | Option 1 (6-Layer) | Option 2 (8-Layer) | Option 3 (10-Layer) |
|--------------------|--------------------|--------------------|---------------------|
| Board Thickness    | 1.0 mm             | 1.2 mm             | 1.6 mm              |
| Material           | FR4                | Low-loss FR4       | Low-loss FR4        |
| Impedance Control  | Yes                | Yes                | Yes                 |
| High-speed Support | PCIe Gen3          | PCIe Gen4          | PCIe Gen5           |
| Cost               | Low                | Medium             | High                |
| Manufacturability  | Easy               | Standard           | Advanced            |



## PCB Floorplanning Intelligence

AI-driven component placement, region planning and thermal optimization

Floorplan  
Agent

## Input: PCB &amp; System Requirements

- Component list (CPU, memory, I/O, etc.)
- Board dimensions and shape
- Connector locations
- High-speed interfaces
- Power delivery requirements
- Thermal constraints
- Mechanical constraints
- EMI/EMC requirements
- Manufacturing constraints
- Keep-out zones

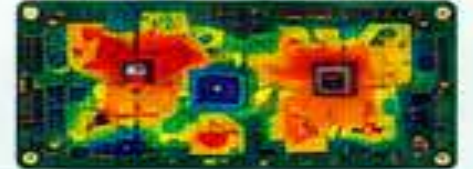


## AI Analysis &amp; Optimization

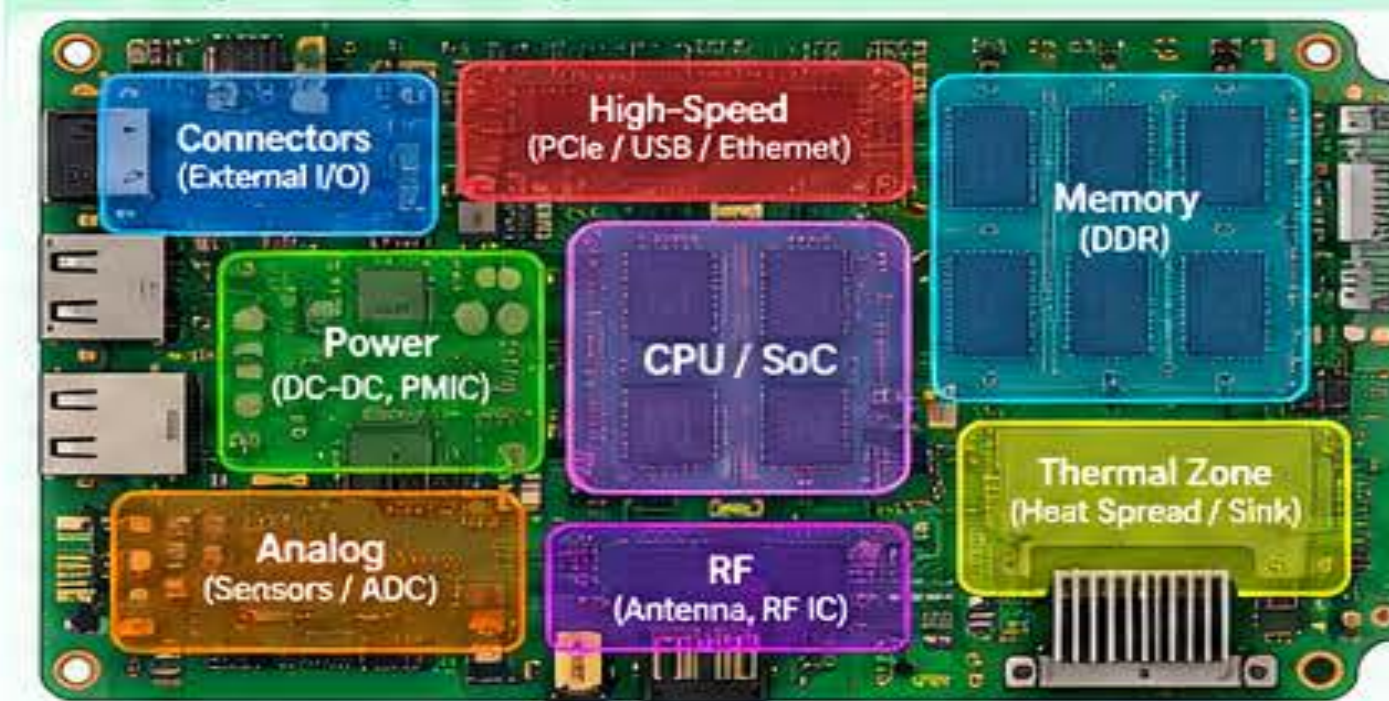
- Identify functional regions
- Optimal component placement
- Connector placement and orientation
- Thermal-aware placement
- High-speed path planning
- Power path planning
- Analog / digital separation
- EMI-sensitive region planning
- Keep-out and mechanical constraints
- Iterative optimization with trade-offs

## Optimized Layout Benefits

- Shorter high-speed paths
- Better thermal performance
- Reduced EMI / crosstalk
- Improved power integrity
- Optimal connector placement
- Analog / digital isolation
- Manufacturing-friendly design
- Easier assembly and test
- Cost effective placement



## AI Floorplanning Example



## Functional Regions

- Connectors**
  - USB, Ethernet, HDMI, etc.
  - Edge clearance
  - Mechanical constraints
- High-Speed**
  - PCIe, USB, Ethernet
  - Length matching
  - Avoid stubs and crosstalk
- CPU / SoC**
  - Central placement
  - Short memory and power paths
- Memory**
  - DDR, LPDDR
  - Minimize trace length
  - Matched routing
- Power**
  - DC-DC, LDO, PMIC
  - Close to load
  - Short, wide power paths
- Analog**
  - Sensors, ADC/DAC
  - Isolated from digital
  - Clean ground region
- RF**
  - Antenna, RF IC
  - Keep-out zone
  - Isolation and shielding
- Thermal**
  - Heat sink, thermal vias
  - Airflow consideration
  - Hotspot analysis

## Floorplanning Optimization Metrics

Thermal  
Analysis

Hotspot reduction



Signal Integrity

Shorter paths  
Less crosstalk

Power Integrity

Optimized power  
paths

EMI/EMC

Noise isolation



Manufacturability

Assembly friendly



Cost Optimization

Optimal placement



Better  
Performance



Lower  
Cost



Faster  
Time to Market



Reliable  
Designs

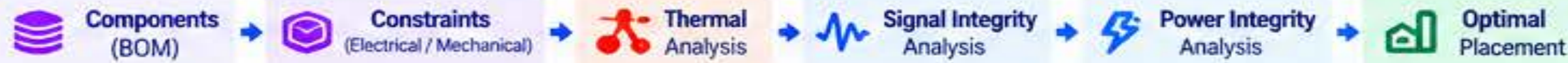


## Placement Intelligence

AI-optimized component placement for performance, thermal and manufacturability



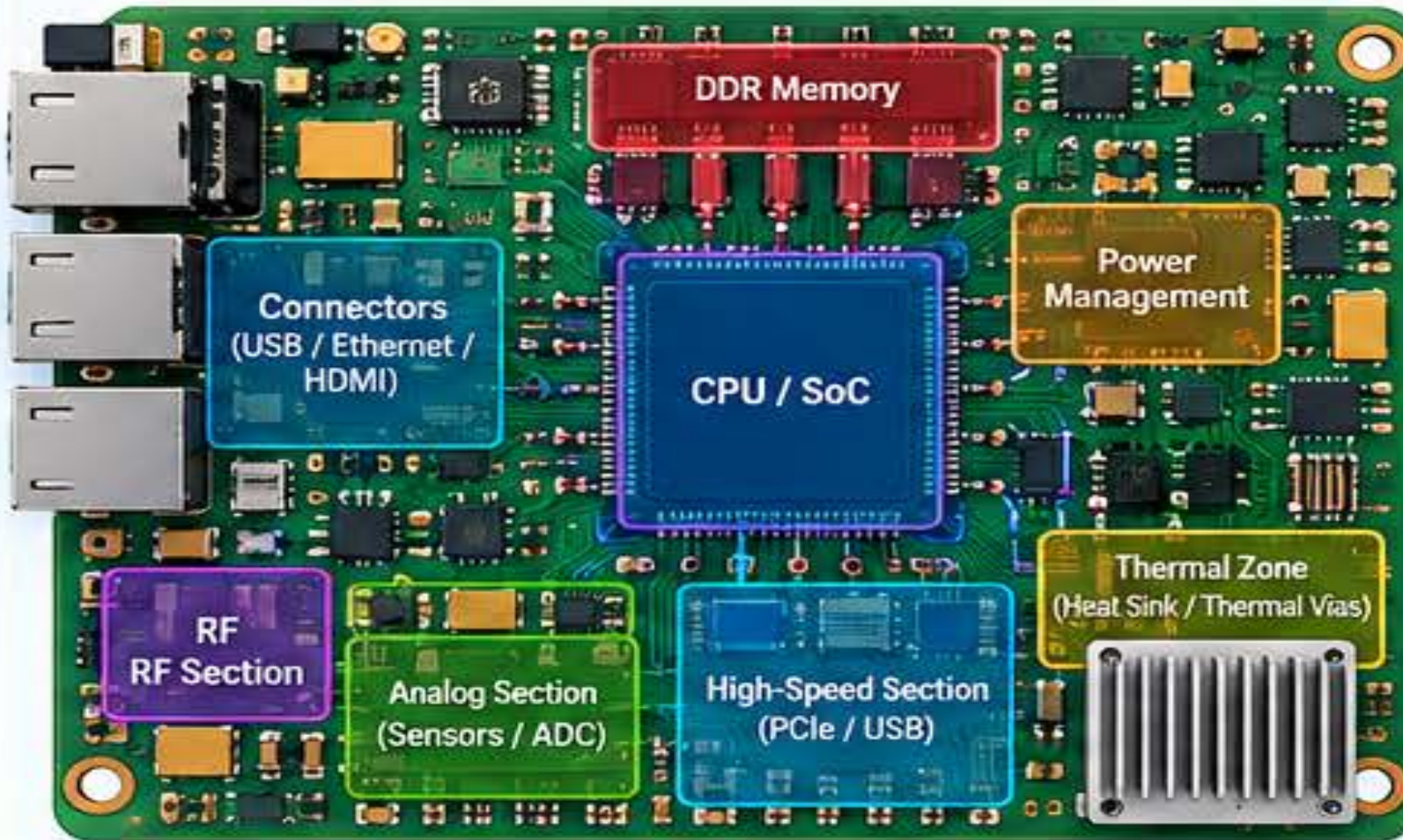
Placement  
Agent



### AI Placement Optimization Flow

- Components**  
From BOM / Netlist
- Constraints**  
Electrical / Mechanical / Keep-out / Rules
- Thermal Analysis**  
Identify hotspots
- Signal Integrity**  
Critical nets proximity
- Power Integrity**  
Power delivery and decoupling
- Mechanical**  
Board shape / Connectors
- Optimal Placement**  
AI-optimized component layout

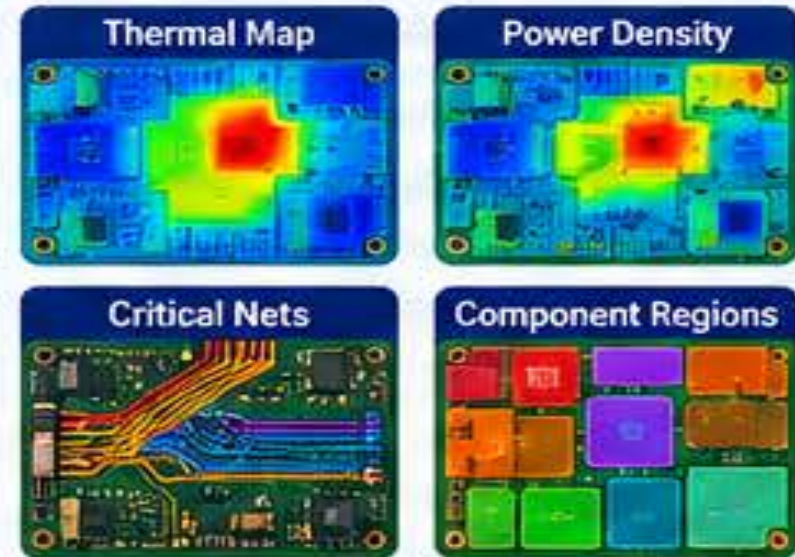
### AI Optimized Placement (Example)



### AI Placement Checks

- ✓ Critical component proximity
- ✓ Decoupling placement
- ✓ Thermal distribution
- ✓ Connector accessibility
- ✓ High-speed path length
- ✓ Clock proximity
- ✓ Power delivery
- ✓ EMI (noise-sensitive components)
- ✓ Manufacturing accessibility
- ✓ Design rule compliance

### Analysis Views



## Routing Intelligence

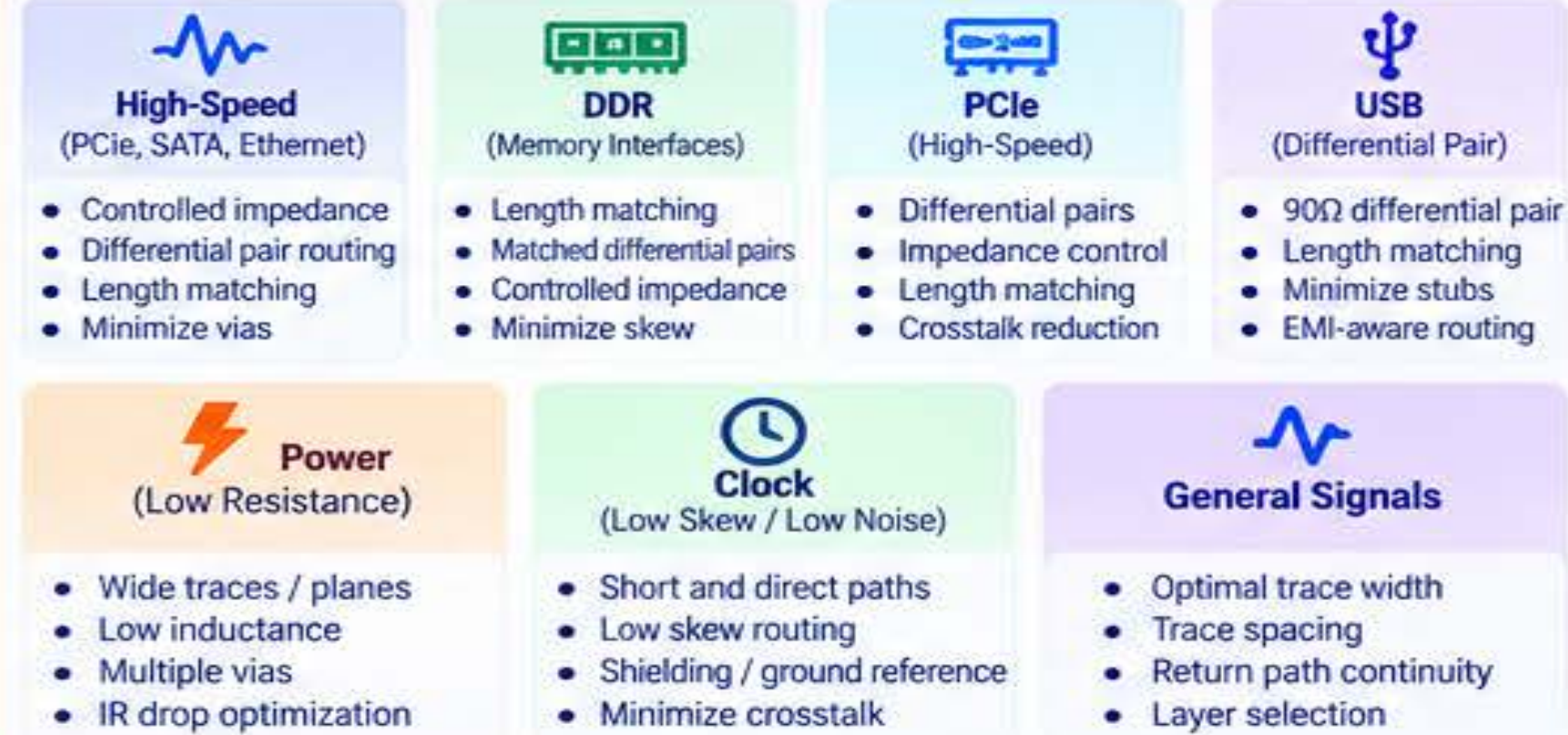
AI-powered intelligent routing for signal, power and high-speed interfaces



Routing  
Agent



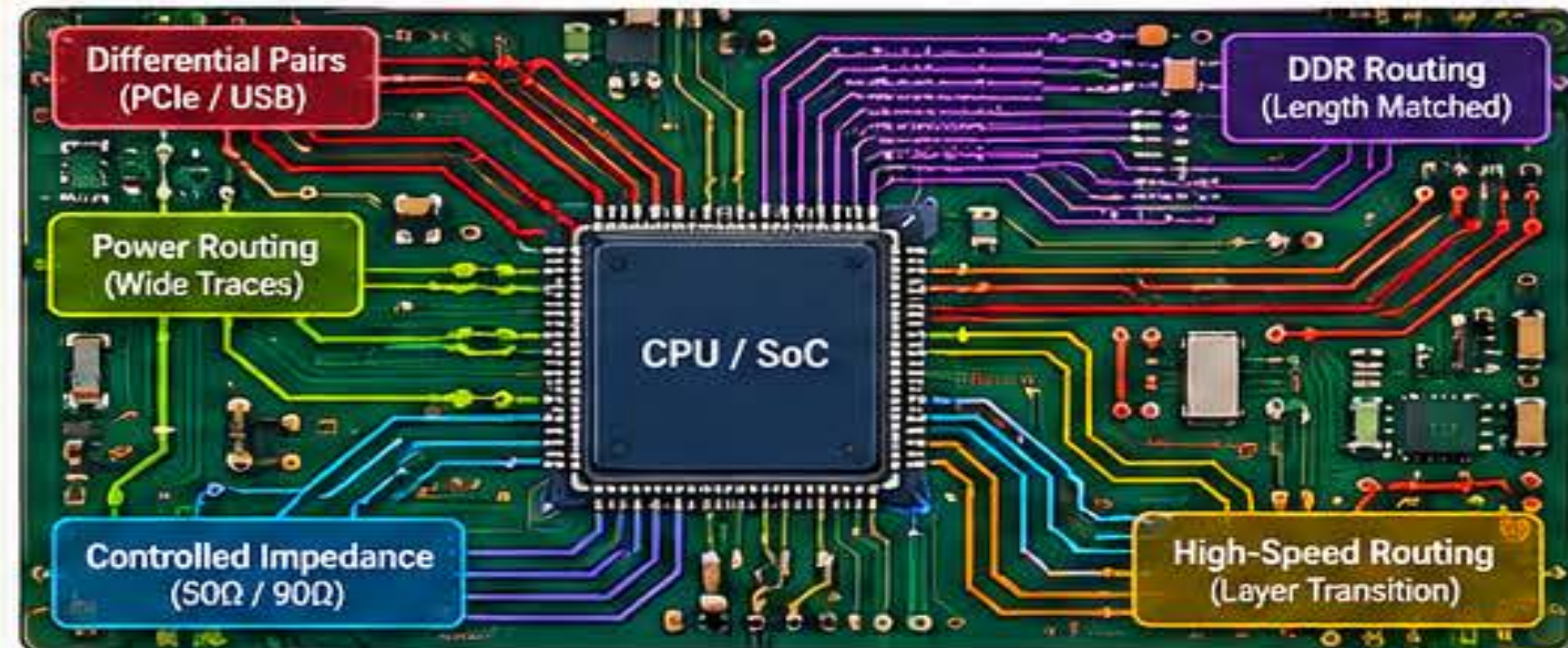
### AI Routing Strategy



### AI Routing Flow

- Netlist**  
From schematic
- Constraints**  
Design rules / Impedance / Clearance / Length
- Critical Nets**  
High-speed / DDR / PCIe / USB / Power / Clock
- Routing Strategy**  
AI optimized routing plan
- Interactive / Automated Routing**  
AI-assisted real-time routing
- Verification**  
DRC, SI/PI, Length match

### AI Intelligent Routing (Example)



### AI Routing Analysis

- ✓ Trace width and spacing
- ✓ Via count and transitions
- ✓ Length matching (DDR/PCIe/USB)
- ✓ Differential pair matching
- ✓ Impedance control
- ✓ Crosstalk analysis
- ✓ Return path and ground reference
- ✓ Via optimization
- ✓ DRC compliance
- ✓ SI/PI verification





Better Performance



Lower Cost



Higher Reliability



Faster Time to Market



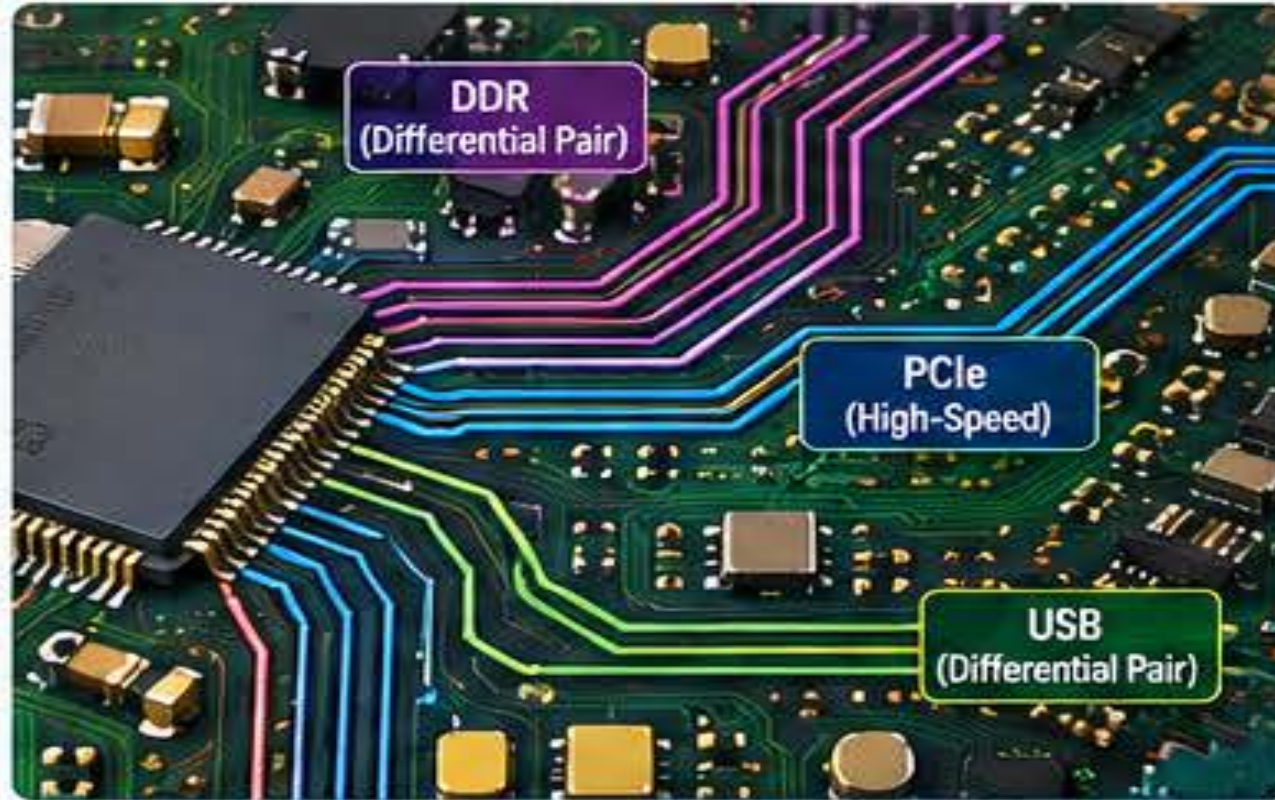
## Signal Integrity Intelligence

Ensure reliable high-speed signal transmission

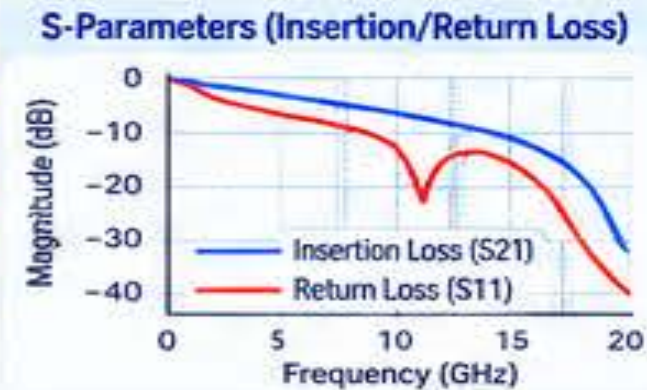
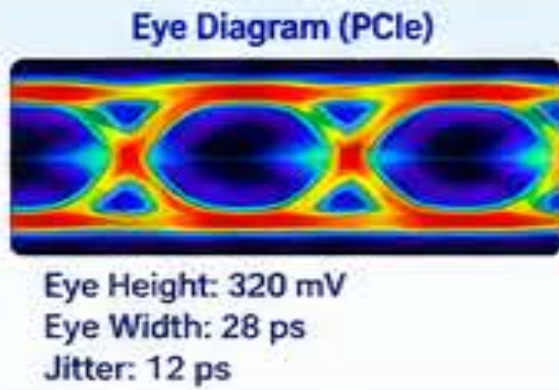
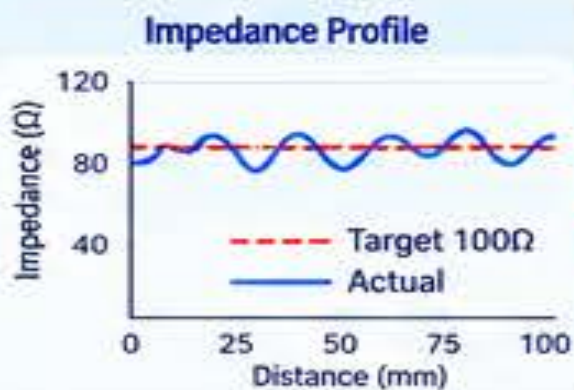


### High-Speed Interfaces

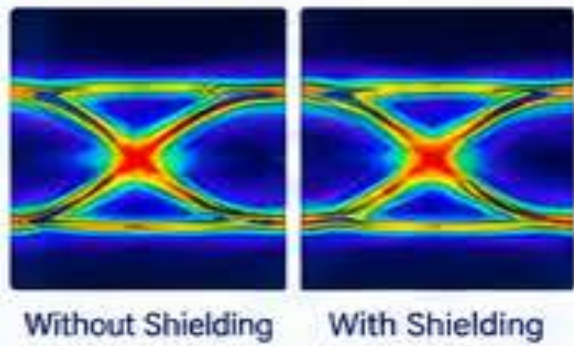
- PCIe
- DDR
- USB
- SATA
- Ethernet
- HDMI / DisplayPort
- MIPI
- SerDes
- Custom High-Speed



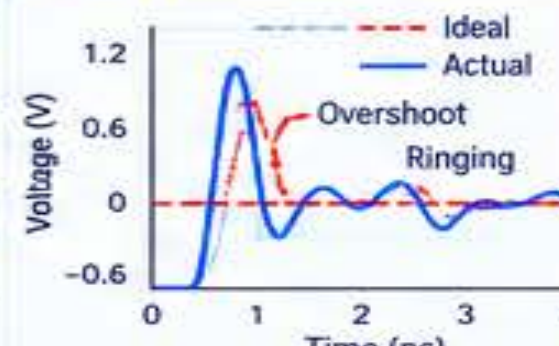
### Signal Integrity Analysis (Examples)



### Crosstalk Analysis



### Time Domain Response



### Signal Integrity Issues Detected

- Reflections (impedance mismatch)
- Crosstalk (near-end / far-end)
- Overshoot / Undershoot
- Ringing
- Insertion loss (high-frequency)
- Return loss (poor matching)
- Timing problems (skew, jitter)
- Eye closure (margin reduction)



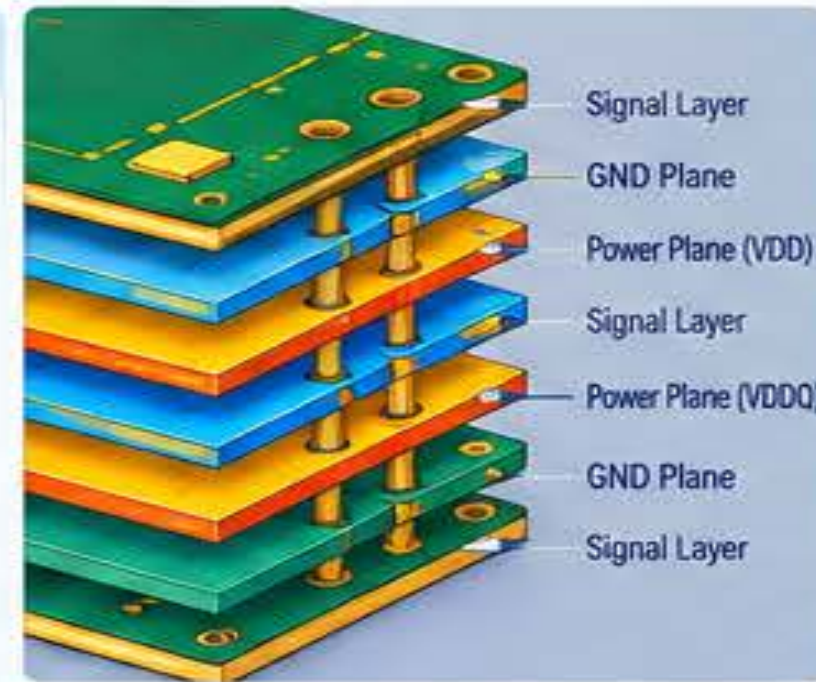
## Power Integrity Intelligence

Stable and clean power delivery for reliable operation



### Power Architecture

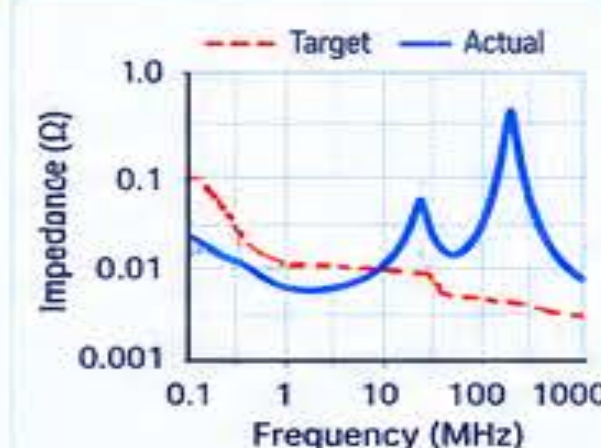
- Voltage Rails (VDD, VDDQ, VPP)
- Current Requirement
- Power Domains
- Decoupling Strategy
- Plane Design (Power/GND)
- Via Strategy
- VRM / PMIC Integration



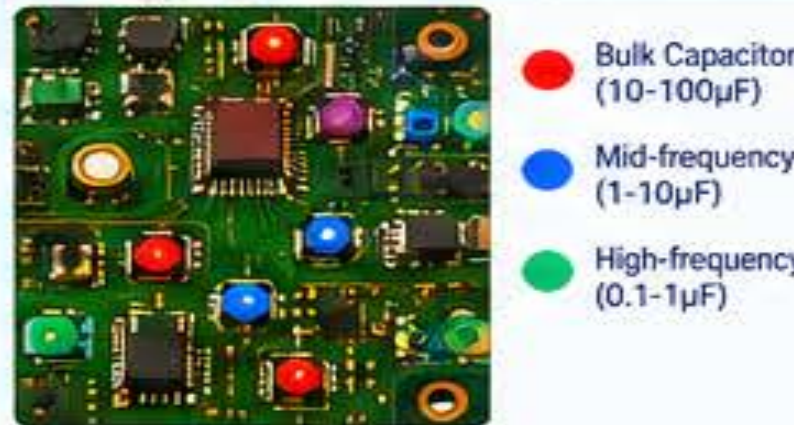
### IR Drop Analysis



### PDN Impedance



### Decoupling Network Analysis



### Power Integrity Checks

- Power plane design
- Current density
- Voltage drop (IR drop)
- Decoupling capacitor selection
- Capacitor placement
- PDN impedance and resonance
- Ground return path
- Power distribution
- Thermal impact on power



## Thermal Intelligence

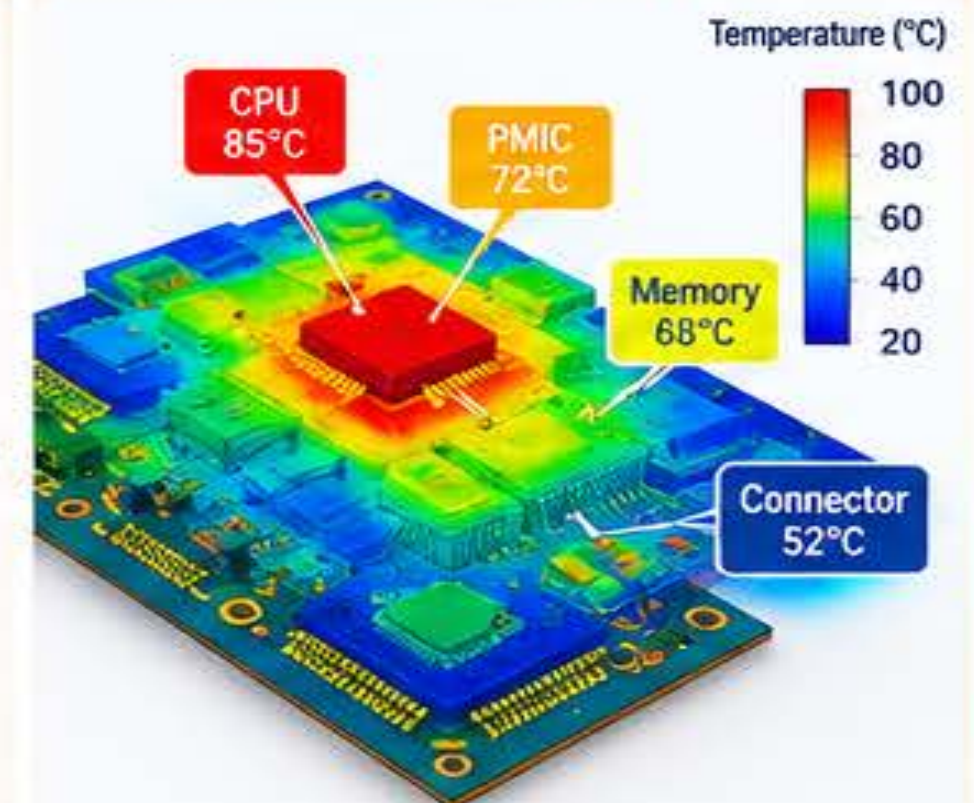
Identify hotspots and ensure thermal reliability



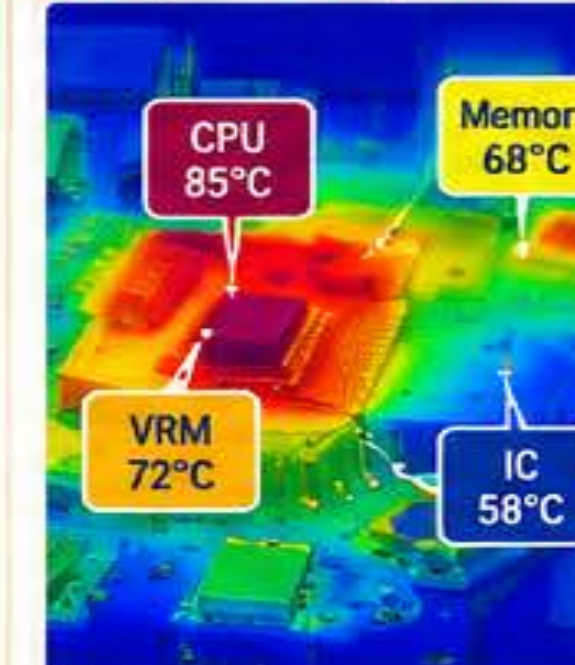
### Thermal Inputs

- Component Power (TDP)
- Ambient Temperature
- Airflow Conditions
- Board Enclosure
- Material Properties
- Heat Sink / Thermal Vias
- Component Placement
- Thermal Constraints

### 3D Thermal Simulation (Example)



### Component Temperature



### Thermal Analysis Checks

- IC temperature estimation
- Thermal hotspots
- Thermal vias and copper area
- Heat sink effectiveness
- Airflow analysis
- Component spacing
- Thermal coupling between components
- Board-level temperature distribution
- Thermal risk assessment





## ERC Intelligence

AI-powered Electrical Rule Checking with Intelligent Explanations



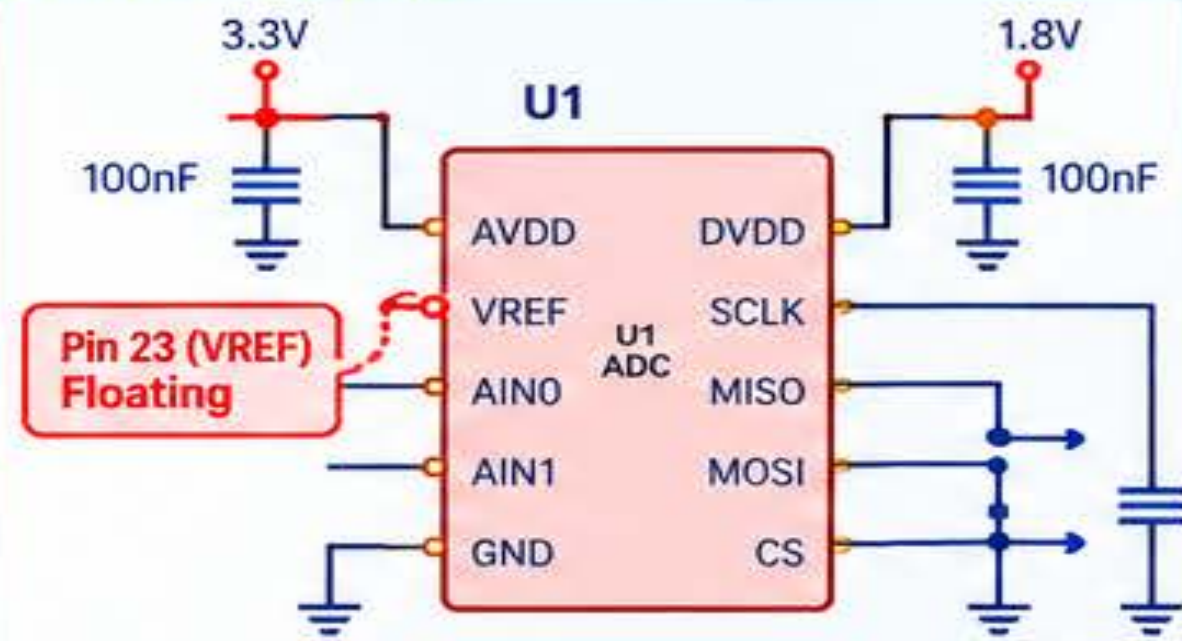
ERC Agent

- Power Pin Checks
- Connectivity Checks
- Voltage & Level Checks
- Input/Output Rules
- Unconnected Pins
- Component Compatibility
- AI Root Cause & Fix

### ERC Rule Checks

- Power Pin (VCC/GND)
- Floating Pin
- Output ↔ Output
- Input ↔ Input
- Missing Power
- Wrong Voltage
- Unconnected Pin
- Bus Width Mismatch
- Pull-up / Pull-down
- Reference Pin (VREF)
- Reset / Clock Rules
- Component Compatibility

### Schematic Example



### ERC Results

Errors 3  
Warnings 5  
Info 2

### AI Explanation (Example)

**ERROR: Pin 23 (VREF) is unconnected.**

#### AI Analysis

Pin 23 is the reference-voltage input (VREF) for the ADC. The datasheet (Rev. E, Section 7.3) recommends a local 100 nF decoupling capacitor and connection to a stable reference voltage (e.g., 2.5V). The current design leaves the pin floating, which can cause inaccurate ADC readings, higher noise and unstable operation.

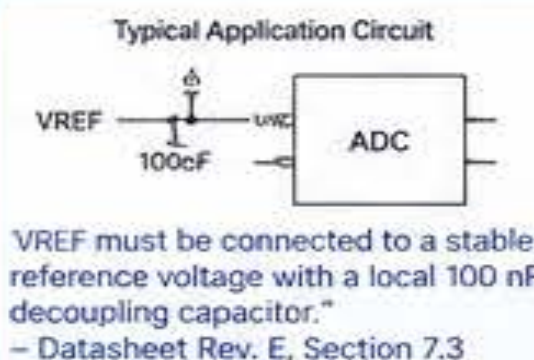
#### Recommended Fix

- Connect Pin 23 to VREF (e.g., 2.5V reference source).
- Add a 100 nF decoupling capacitor close to the pin.
- Use short trace with ground reference.
- Verify reference voltage accuracy and noise.

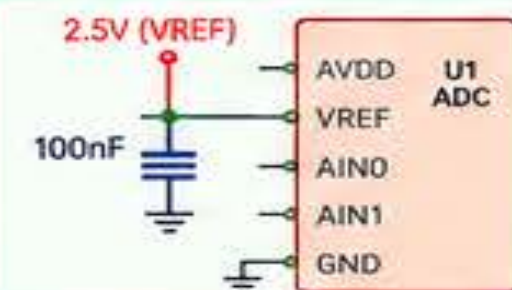
#### Impact if Not Fixed

- Unstable ADC operation
- Incorrect measurements
- Increased noise and possible system failure

### Datasheet Reference



### Fixed Design (AI Suggestion)



## DRC Intelligence

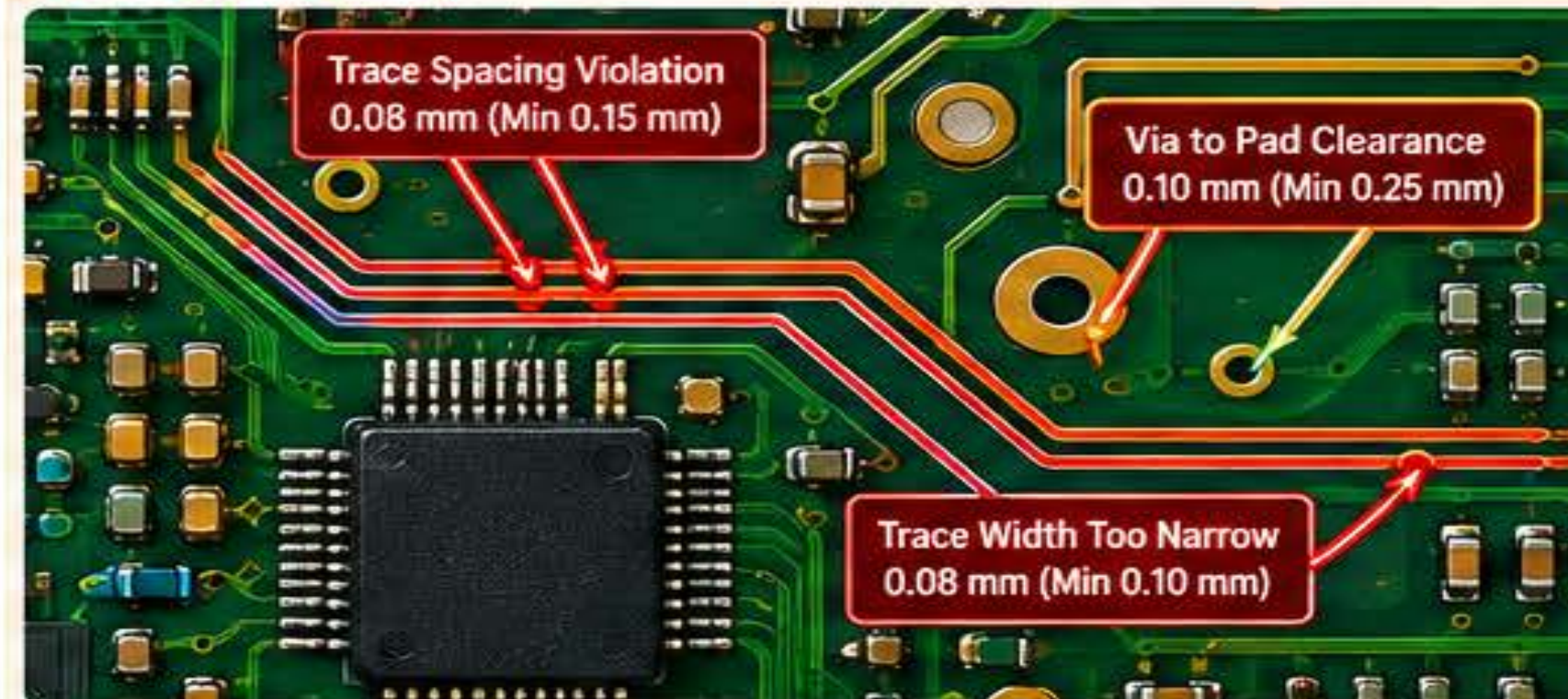
AI-powered Design Rule Checking with Root Cause Analysis and Intelligent Fixes



DRC Agent

- Clearance
- Trace Width
- Via Rules
- Hole Size
- Spacing
- Edge to Copper
- Layer Rules
- Solder Mask
- Manufacturing Rules
- AI Root Cause & Fix

### PCB Layout Example



### DRC Results

Errors 8  
Warnings 12  
Info 5

### DRC Rule Checks

- Clearance (Net to Net)
- Trace Width
- Via Size / Drill Hole
- Annular Ring
- Spacing (Pad to Pad, Via to Via)
- Edge Clearance
- Layer Violations
- Solder Mask
- Silkscreen
- Manufacturing Constraints (DFM)

### AI Explanation (Example)

**ERROR: Trace spacing violation (0.08 mm, required  $\geq 0.15$  mm).**

#### AI Root Cause Analysis

- These two traces (DDR\_DQ[3] and DDR\_DQ[4]) are too close.
- Spacing is 0.08 mm, while the board rule requires 0.15 mm.
- The violation is caused by manual routing in a congested area near the DDR memory.
- These are high-speed DDR signals, so inadequate spacing can increase crosstalk.

#### Impact Analysis

- Increased crosstalk and signal integrity issues
- Potential DDR read/write errors
- Timing margin reduction
- May fail at higher data rates

#### Recommended Fixes

- Increase spacing to  $\geq 0.15$  mm.
- Reroute using interactive router (AI-assisted).
- Move one trace to adjacent layer with ground reference.
- Use differential pair spacing rules if applicable.
- Re-run DRC to ensure no new violations.

#### Before vs After (AI Suggestion)



### AI Verification Flow



### Key Benefits

- Actionable Insights: Not just errors, but explanations
- Faster Debug: AI root cause analysis
- Better Design Quality: Fewer iterations
- Reduced SI/PI Issues: Early detection
- Manufacturing Ready: DFM-aware checks
- Lower Cost: Avoid respins
- Reliable Products: Higher first-pass success



Better Performance



Lower Cost



Manufacturing Ready



Faster Time to Market



## EMI/EMC Intelligence

AI-powered EMI/EMC analysis, noise prediction and mitigation



EMI/EMC Agent



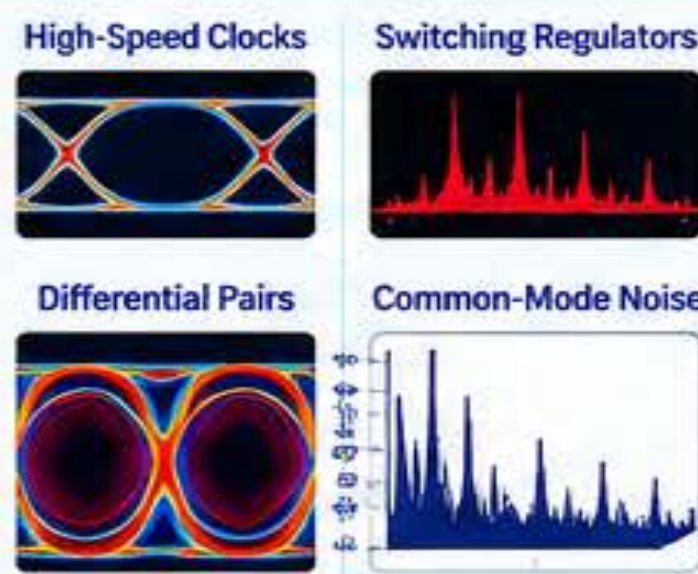
### EMI/EMC Analysis Inputs

- PCB Layout (Gerber/ODB++)
- Stackup & Layer Information
- Component Models
- High-Speed Interfaces
- Power Architecture
- Clock & Switching Signals
- Grounding & Return Paths
- Enclosure / Shielding
- Compliance Standards (CISPR, FCC, CE)

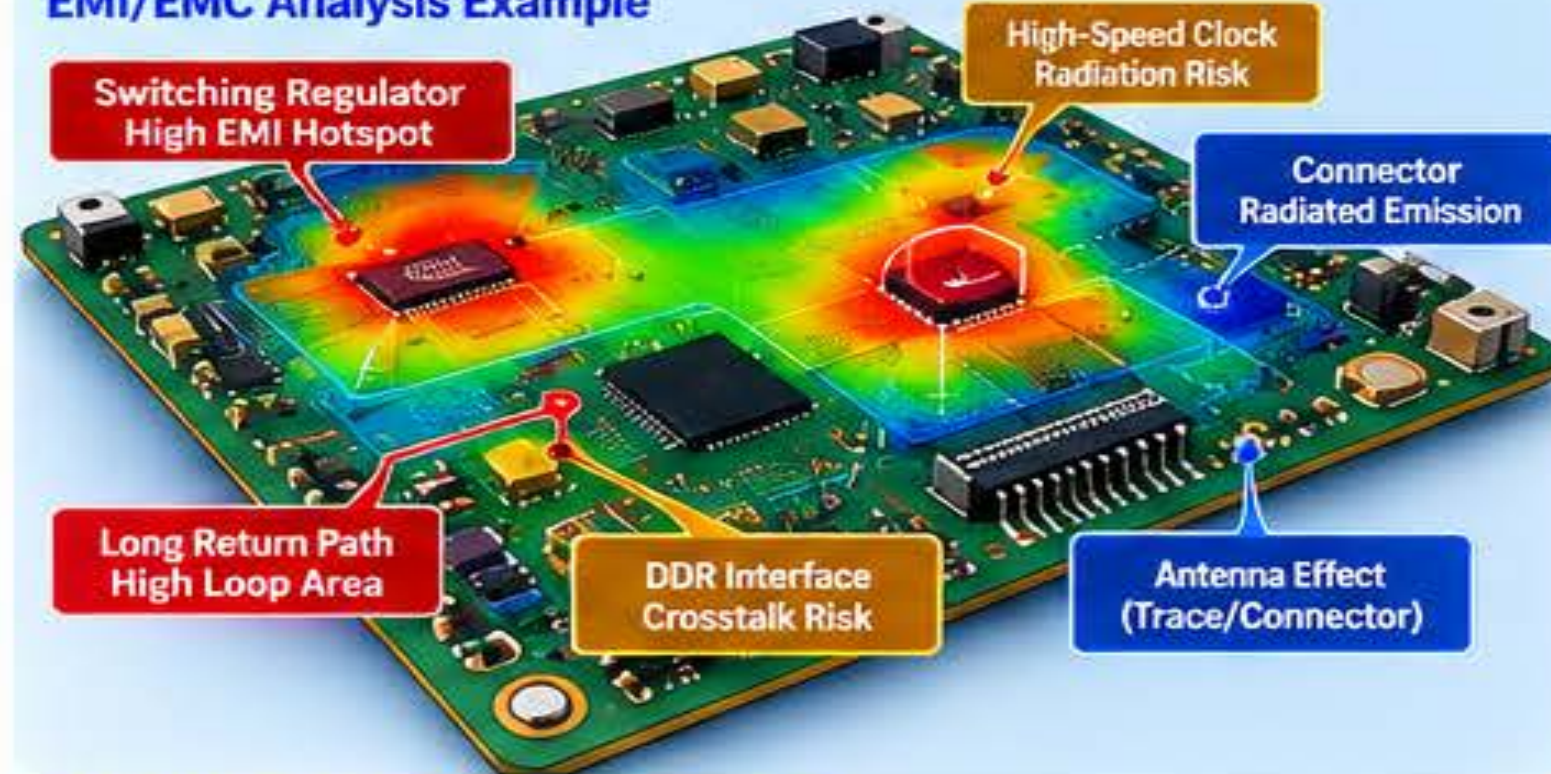
### AI Analysis and Insights

- Identify high-speed clocks and critical nets
- Analyze switching regulators and harmonics
- Detect return path discontinuities
- Calculate loop areas and radiation risk
- Find ground and power plane issues
- Analyze coupling paths (conducted & radiated)
- Identify common-mode noise sources
- Evaluate shielding effectiveness
- Recommend filtering and mitigation
- Check compliance with EMI/EMC standards

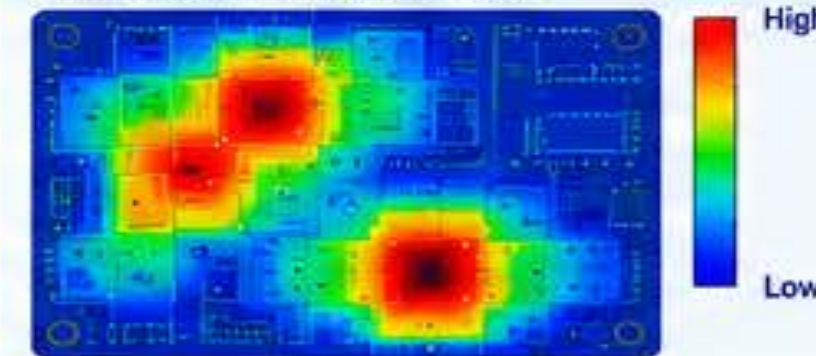
### Noise Sources



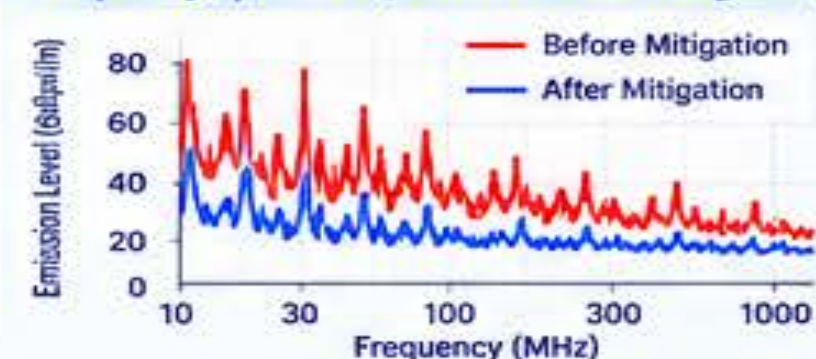
### EMI/EMC Analysis Example



### Field Emission Map (Simulated)



### Frequency Spectrum (Before vs After Mitigation)



### EMI/EMC Mitigation Recommendations

- Optimize Return Paths
- Reduce Loop Areas
- Add Filtering (Ferrite, LC)
- Use Differential Routing
- Improve Shielding (Enclosure, GND)
- Optimize Stackup (Plane Continuity)
- Check Regulatory Compliance



## Manufacturing Intelligence

AI-powered manufacturing verification and release package validation



Manufacturing Agent



### Manufacturing Inputs

- Schematic
- PCB Layout
- Gerber Files
- ODB++ Files
- Drill Files (NC Drill)
- Pick & Place Files
- BOM (Bill of Materials)
- Assembly Drawings
- Component Datasheets
- Design Rules / Constraints
- Manufacturer Capabilities (DFM Rules)

### AI Verification Checks

- Schematic ↔ PCB consistency
- PCB ↔ Gerber consistency
- Netlist and component verification
- BOM completeness and correctness
- Pick & Place accuracy
- Assembly drawing verification
- Drill file validation
- Layer stack and Gerber integrity
- Solder mask and silkscreen checks
- DFM (Design for Manufacturing) checks
- Component availability and alternates
- Manufacturing capability validation
- Final release package verification

### Manufacturing Output

- Verified Gerber Files
- Validated ODB++ Files
- Checked Drill Files
- Verified Pick & Place Files
- Complete and Correct BOM
- Assembly Drawings
- DFM Report
- Manufacturing Readiness Report
- Component Availability Report
- Release Package (ZIP)

### Manufacturing Benefits

- Fewer Manufacturing Errors
- Lower NRE Cost
- Faster Time to Production
- Higher First-Pass Yield

### Release Package Verification Flow



### DFM Analysis Example





<https://pragyanai-pcb-varification-solutions-poc.streamlit.app/>

[https://github.com/pragyanaischool/PragyanAI\\_PCB\\_Verification\\_Solution](https://github.com/pragyanaischool/PragyanAI_PCB_Verification_Solution) Share ☆ ✎ 🔄 ⋮

App Settings

Analysis Mode

Full (Accurate) ▼

- ☒ Show Raw Outputs
- ☒ Cleanup temp files

i Info

- Multi-Agent AI
- Power / Signal / Thermal / Layout
- Vision + OCR + Graph + GNN

PragyanAI PCB Copilot



Grow with Gyan

Multi-Agent AI for PCB Analysis (Vision + Graph + LLM + GNN)

Upload PCB Image or Netlist

 buggy\_amp.txt × +  
353.0B

Supported Inputs

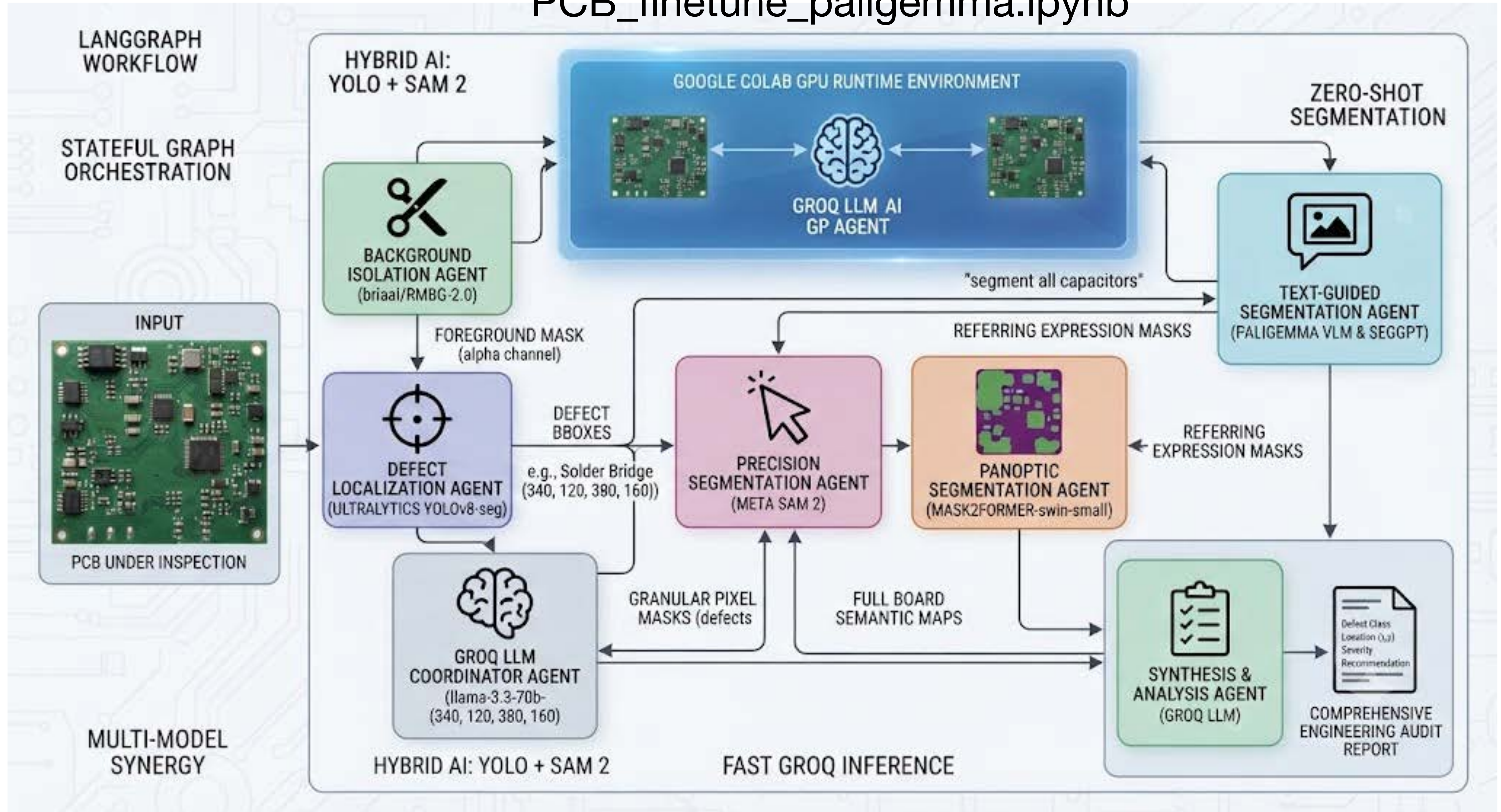
- PCB Images
- Netlist Files

Run AI Analysis

Analysis Completed

# PCB\_Related Projects.ipynb

## PCB\_finetune\_paligemma.ipynb



# PCB\_Related Projects.ipynb

<https://github.com/pragyanaischool/PragyanAI-PCB-Varifications-Solutions>

### Settings

Analysis Mode

Full (Accurate) ▾

☒ Show Debug Info

### PCB Verification Stack

- Vision Defect Localization
- IPC-A-610 Rule Checking
- Topological Graph Engine
- Multi-Agent Orchestration

### Chat Controls

Clear Chat

Reset Memory

Show Memory

Share ☆ ✎ ↺ ⋮

## PRAGYAN AI

Grow with Gyan

# PragyanAI PCB Copilot

Vision + YOLO + Segmentation + Multi-Agent AI + RAG

Upload PCB Image

 PCB\_damaged.jpeg 47.0KB +

Upload Datasheet / Spec PDF (Optional)

 Upload 200MB per file • PDF



# PCB\_Related Projects.ipynb

<https://github.com/pragyanaishchool/PragyanAI-PCB-Verifications-Solutions>

### Settings

Analysis Mode

Full (Accurate) ▾

☒ Show Debug Info

---

### PCB Verification Stack

- Vision Defect Localization
- IPC-A-610 Rule Checking
- Topological Graph Engine
- Multi-Agent Orchestration

### Chat Controls

Clear Chat

Reset Memory

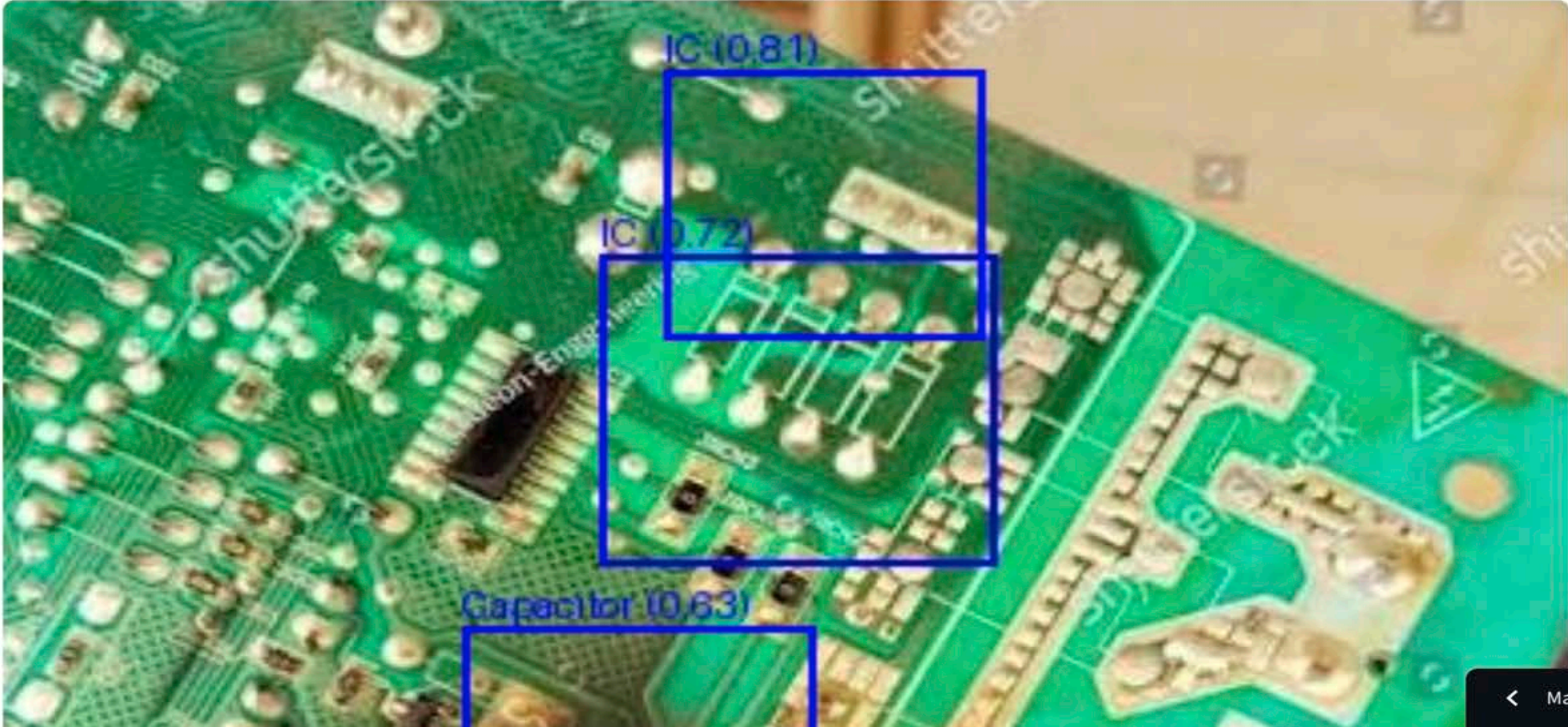
Show Memory

Share ☆ ✎ ↺ ⋮

1. Final Report 2. Visualization 3. Vision AI 4. Insights 5. Domain Rules 6. Graph Summary 7. Debug 8. Chat AI

## PragyanAI PCB Visualization

☒ Show Components ☒ Show Issues ☒ Show Heatmap



Manage app

<https://pragyanai-pcb-part1-planning-engine-solution-ah7xd8irnzed5wkym.streamlit.app/>

<https://github.com/pragyanaishchool/PragyanAI-PCB-Part1-Planning-Engine-solution>



Grow with Gyan

Planning Engine

LLM: openai/gpt-oss-120b (Groq)

State Machine: LangGraph

Embeddings: HF all-mpnet-base-v2

Reset Session



# PragyanAI Element 1: Planning & Strategy

Transform natural language PRDs into validated `architecture_plan.json` files.

## Input: Product Requirements (PRD)

Describe your hardware project in detail:

A solar-powered IoT weather station with BME280 sensor, ESP32-S3, and an e-paper display. Must operate at 3.3V.

## Output: Architecture Plan

```
{
 "mcu" : {
 "family" : "ESP32-S3"
 "package" : "ESP32-S3-WROOM-1"
 }
 "power_tree" : [
```

<https://pragyanai-frontenddesignspecification.streamlit.app/>

[https://github.com/pragyanaischool/pragyanai\\_eda\\_aiagentic\\_vlsi-front-end-design-specifications\\_sol/](https://github.com/pragyanaischool/pragyanai_eda_aiagentic_vlsi-front-end-design-specifications_sol/)

<<

app

Knowledge Base

Spec Designer

Spec Chat

RTL Generator

Verification Bench

PRAGYAN

AI

Grow with Gyan

Share ☆ ↗ ↻ ⋮

PRAGYAN

AI

Grow with Gyan

PragyanAI: Multi-Agent VLSI Design Suite

Autonomous Chip Design Framework with RAG, RTL Gen & HITL

Welcome to **MAGE** (Multi-Agent General Engine) tailored for VLSI Front-End Design. This platform automates the entire lifecycle—from **RTL Design Specifications (RDS)** to **Synthesizable Code** and **Self-Checking Testbenches**—by leveraging specialized AI agents and your local technical library.

👉 Select a page from the sidebar to begin your design journey.

<https://pragyanai-frontenddesignspecification.streamlit.app/>

[https://github.com/pragyanaishool/pragyanai\\_eda\\_aiagentic\\_vlsi-front-end-design-specifications\\_sol/](https://github.com/pragyanaishool/pragyanai_eda_aiagentic_vlsi-front-end-design-specifications_sol/)



## 1. Knowledge Base

Upload your PDF standards (AXI, PCIe, RISC-V, etc.). The **RAG Engine** indexes these locally using HuggingFace embeddings for precise technical grounding.

## 2. Spec Designer

Enter a problem statement. The **Architect** drafts the RDS, the **Critic** audits for CDC/Protocol errors, and you provide **Human-in-the-Loop** refinements to create the "Golden Spec".

## 3. Spec Chat

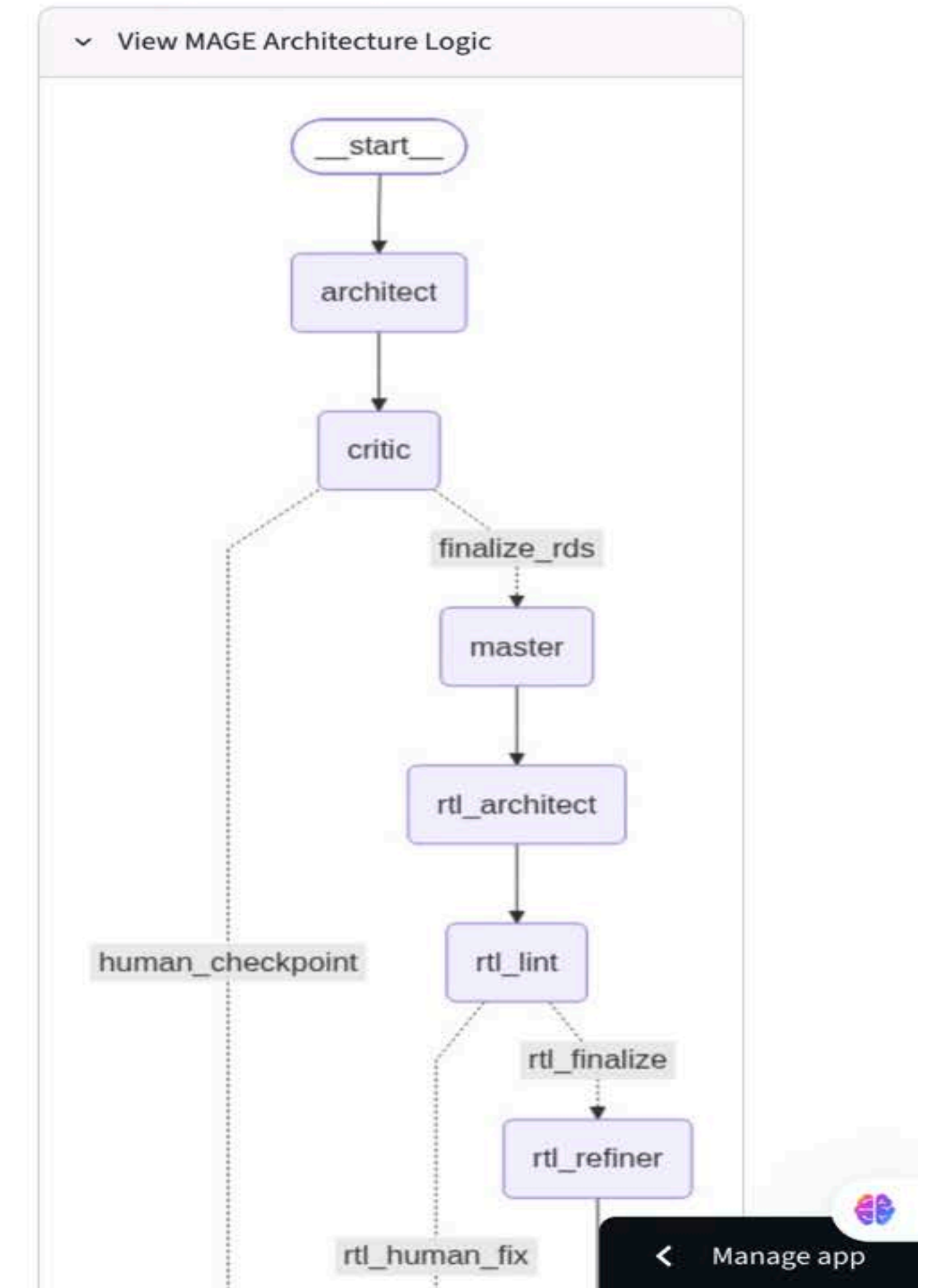
Interact with your generated specification. Ask questions about design choices or protocol nuances found in your uploaded documents.

## 4. RTL Generator

Translate your RDS into production-ready **Verilog** or **VHDL**. Includes an automated **Linting Agent** to catch "design killer" bugs like unintentional latches.

## 5. Verification Bench

Generate a **Self-Checking Testbench**. This phase includes a **DV Lead Agent** that explains the rationale behind every test case (Corner cases, Reset, and Stress tests).



<https://pragyanai-frontenddesignspecification.streamlit.app/>

[https://github.com/pragyanaischool/pragyanai\\_eda\\_aiagent\\_vlsi-front-end-design-specifications\\_sol/](https://github.com/pragyanaischool/pragyanai_eda_aiagent_vlsi-front-end-design-specifications_sol/)

app

Knowledge Base

Spec Designer

Spec Chat

RTL Generator

Verification Bench

# RTL Design Specification (RDS) – 32-Bit Pipelined Adder

Document Revision: 1.0 – 2026-09-25  
Author: Principal VLSI Architect

## 1. Functional Overview

| Item                    | Description                                                                                                                                                                                                                                                                                                                                                |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Block name              | adder32_pipe                                                                                                                                                                                                                                                                                                                                               |
| Purpose                 | Provide a high-throughput, fully synchronous 32-bit unsigned (or two’s-complement) addition service to the SoC. The block is deliberately pipelined to achieve one-addition-per-clock-cycle throughput after pipeline fill, while keeping the combinational delay of the adder itself well within a single clock period.                                   |
| SoC integration context | The adder is a shared arithmetic accelerator used by multiple masters (e.g., CPU, DMA, DSP). It is accessed through a memory-mapped register interface on the APB/AXI-Lite bus. The design follows the standard <b>Peripheral Register Interface (PRI)</b> used across the SoC, enabling easy inclusion in the address decoder and power-management block. |

# Technical Audit – RTL Design Specification (RDS) “32-Bit Pipelined Adder”

Document Reviewed: RDS Draft v1.0 (2026-09-25)  
Auditor: Senior DV Engineer  
Date: 2026-09-25

## 1. Executive Summary

The specification describes a clean, two-stage, fully-synchronous 32-bit adder intended for APB/AXI-Lite access. The high-level architecture is sound, but several **design-killer** risks are either implicit or omitted:

| Area                             | Severity | Comments |
|----------------------------------|----------|----------|
| <div><div>Manage app</div></div> |          |          |

<https://pragyanai-siliconai-ip-dataset-rag-assistant.streamlit.app/>

Share ☆ ✎ ↺ ⋮



# Grow with Gyan

## IP Core & Knowledge Base Management (SQL + FAISS CPU)

Organize semiconductor IPs stored persistently in SQLite database, manage added documents, ingest multi-source data into FAISS CPU Vector Store, and retain information across sessions.

### 1. Select Existing Group / Protocol Family or Create New

Protocol Group Management

Select Existing Group ▼

IP Model / Version Identifier

e.g., PCIe\_Gen2, AXI4\_Interconnect

IP Group / Protocol Family

e.g., PCIe, AMBA, SPI, RISCv

Register / Set Active IP Container

< Manage app



### PragyanAI Silicon RAG

Hardware IP & Datasheet Engineering Suite

#### Navigation Menu

- ☒ Ingestion & IP Management
- ☐ Interactive IP RAG Chat
- ☐ References & Citation Dashboard
- ☐ Spec & Version Comparison
- ☐ Document Viewer & Page Summarizer

#### Registered IP Families

No IP models registered yet. Go to Ingestion to start.

<https://g2tuiszyrge4wvg2ky9bgz.streamlit.app/>

[https://github.com/pragyanaischool/PragyanAI\\_EDA\\_PCB\\_Part2\\_Implementation\\_Core\\_Solutions/](https://github.com/pragyanaischool/PragyanAI_EDA_PCB_Part2_Implementation_Core_Solutions/)

app

Blueprint

Synthesis

Intelligence

Visualizer

PRAGYAN AI

Grow with Gyan

PragyanAI Studio

Select Workspace:

☒ Phase 1: Architecture & Critic

☐ Phase 2: PCB EDA & Defect Prediction

Phase 1: Architecture & RAG Critic

Establish the logical foundation and validate via AI Reviewer.

Share ☆ ↻ ⋮

PRAGYAN AI

Grow with Gyan

Phase 1: AI Architect & Critic

Welcome to the **PragyanAI Implementation Core**. This phase facilitates the **Human-in-the-Loop** transition from planning to synthesis. Review your Architecture Plan, run the **RAG Critic**, and finalize the logic before generating engineering artifacts.

Project Name

e.g., Smart AgTech Sensor Hub V2

Upload Architecture JSON

Upload

200MB per file • JSON

Manage app

https://g2tuiszyrge4wvg2ky9bgz.streamlit.app/

https://github.com/pragyanaischool/PragyanAI\_EDA\_PCB\_Part2\_Implementation\_Core\_Solutions/

PragyanAI Studio

Select Workspace:

Phase 1: Architecture & Critic

Phase 2: PCB EDA & Defect Prediction

Phase 2: Core Solutions EDA

Analyze production line telemetry and evaluate real-time defect risk.

Upload Inspection Data (CSV)

Upload

200MB per file • CSV

Agentic Pipeline Status:

Architect Agent: Online

Critic Agent: RAG-Enabled

Share ☆ ↻ ⋮

PRAGYAN AI

Grow with Gyan

PCB Inspection & Defect Risk Analytics

Automated exploratory telemetry analysis and predictive defect assessment.

Total Inspected Boards

Defect Rate

Defective Units

Mean Reflow Temp

1,500

20.80%

312

244.7 °C

↑ 5.8% vs baseline

Interactive EDA

Model Diagnostic

Live Defect Predictor

Process Telemetry vs. Defect Rate

Manage app