2026 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SANTA CLARA, CA, USA
MARCH 2 - 5, 2026

Real-Time Performance Insights: AI-Accelerated Trace-Driven Analysis for Multi-GPU Pre-Silicon Verification Environments

Vinoth Selvan and Prashanth Rajan

NVIDIA

Agenda

Motivation	→	*Traditional debug doesn't scale*
AI-Accelerated Flow	→	*The paradigm shift*
Tracker Infrastructure	→	*The data foundation*
Analysis Engines	→	*Purpose-built tools*
PerfInsights MCP	→	*Natural language meets DV*
Case Study	→	*Real bugs, rapid root causes*
Results	→	*The impact*
Future Vision	→	*The roadmap*

Key Insight: *"From waveform bottlenecks to AI-powered root cause - the debug evolution in 25 minutes"*

Motivation

Performance analysis is challenging

Bottleneck detection requires intense debug

Traditional debug methodologies are time and resource intensive

Waveform analysis often involves RTL and DV engineers

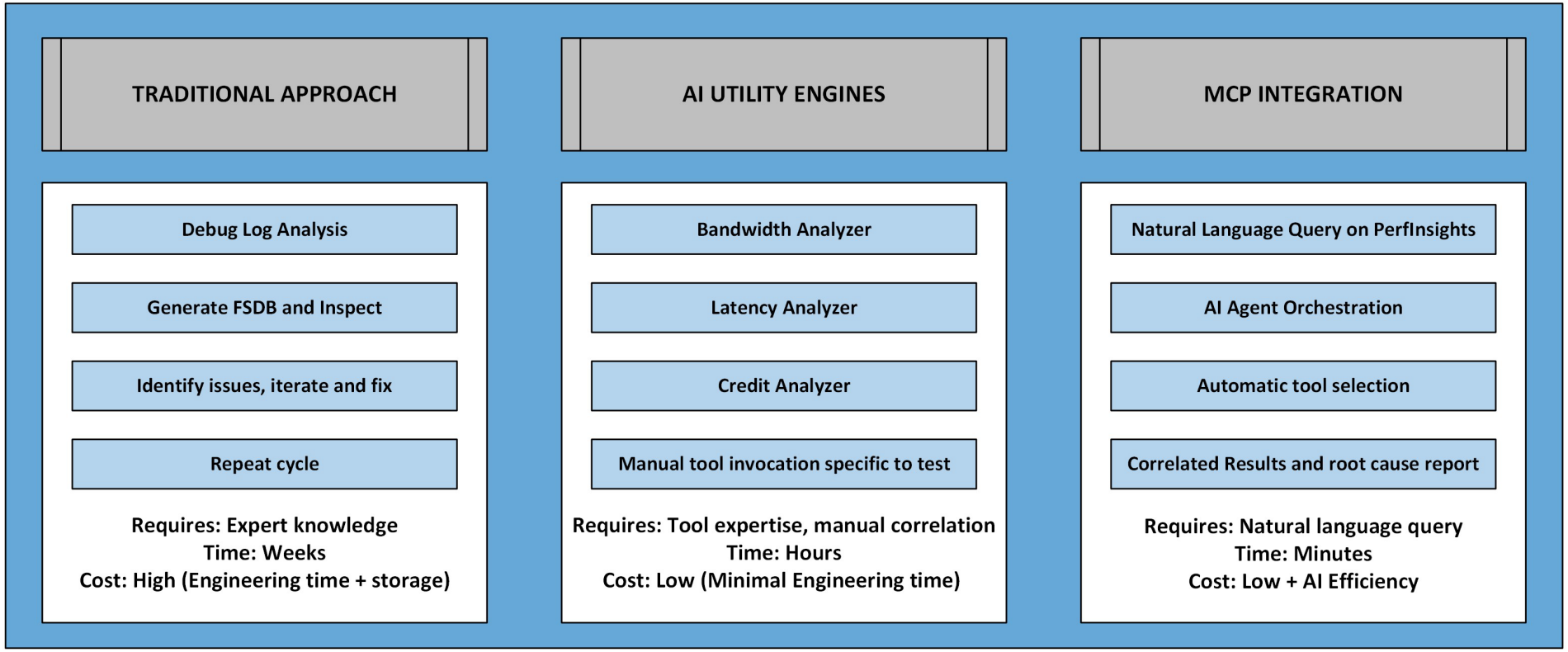
Storage cost + iteration cost + missed schedule

Impact: Repeated waveform debug, pending analysis backlogs, lower confidence in performance health, late-stage bug detection

Solution: Trace-driven multi-tool AI analysis, faster one-time debug, eliminate waveform dump, early-bug detection, 100% confidence

Key Insight: *"From weeks of waveform hunting to minutes of root cause delivery - No storage. No waiting. No exceptions"*

AI-Accelerated Flow – The Journey



Key Insight: *"Each stage removes a bottleneck: FSDB overhead → tool invocation complexity → cross-domain correlation"*

Movement to an AI-Driven Flow

Debug Log Analysis → Generate FSDB → Inspect Waveforms → Identify Issues → Iterate → Repeat (/test)

- **Manual** - every step requires expertise
- **Sequential** - one issue at a time, no parallelism
- **Expert-dependent** - deep architectural knowledge required

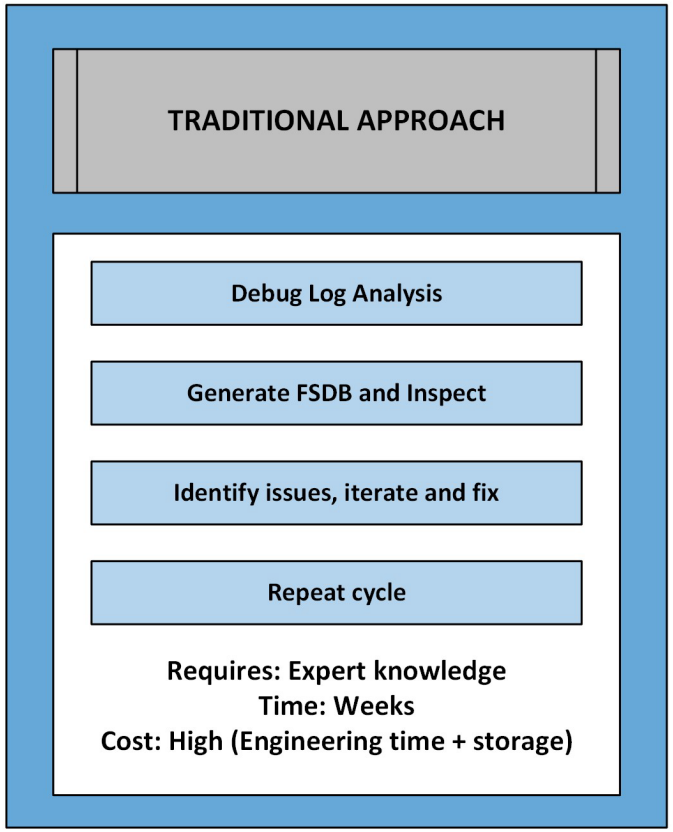
The Cost

- **Compute** – 2 to 4 weeks to generate full-chip FSDB
- **Storage** - 500 GB to 2TB per regression
- **Engineering** - Days of waveform inspection per issue
- **Coverage** - Only ~10% of FSDB analysis complete

The Consequence

- 90% of tests re-run without performance analysis → Debug backlog grows faster than capacity
- **Late-stage bug discovery** - issues found in silicon, not simulation
- **Schedule risk** - debug becomes the longest pole in the tent

Traditional: The FSDB Bottleneck (Weeks)



Key Insight: *"Debug is the bottleneck that doesn't parallelize"*

Movement to an AI-Driven Flow

The Shift : Trace Logs → Purpose-Built Analyzers → Targeted Insights → Manual Correlation

- **Trace-driven** - no waveform generation, no storage explosion
- **Specialized tools** - each engine solves a specific question
- **Scriptable** - automated invocation, repeatable results

The Gain

- **Storage** - 2GB to 200GB per regression,
- **Generation** - Real-time , **Analysis Time** – Hours , **Coverage** - 100% of tests, but manual

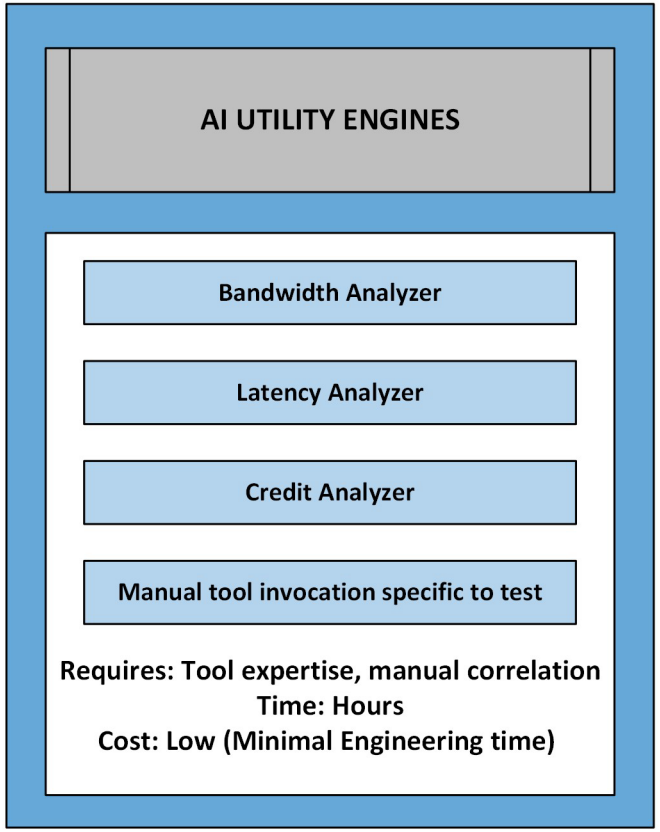
What's Still Missing

- You still choose the tool - which analyzer for this symptom?
- You still invoke manually - command-line expertise required
- You still correlate results - bandwidth + credits + latency => infer conclusion

The Remaining Bottleneck

- Tool selection and cross-correlation remain manual

AI Engines: Eliminate FSDB (Hours)



Key Insight: *"Storage bottleneck solved ! Tool orchestration: still manual"*

Movement to an AI-Driven Flow

The Transformation:

Natural Language Query → AI Orchestration → Multi-Tool Analysis → Correlated Root Cause

- **Conversational** - describe the problem in plain English
- **Autonomous** - AI selects tools, sequences analysis, correlates findings
- **Intelligent** - cross-domain knowledge integration in one response

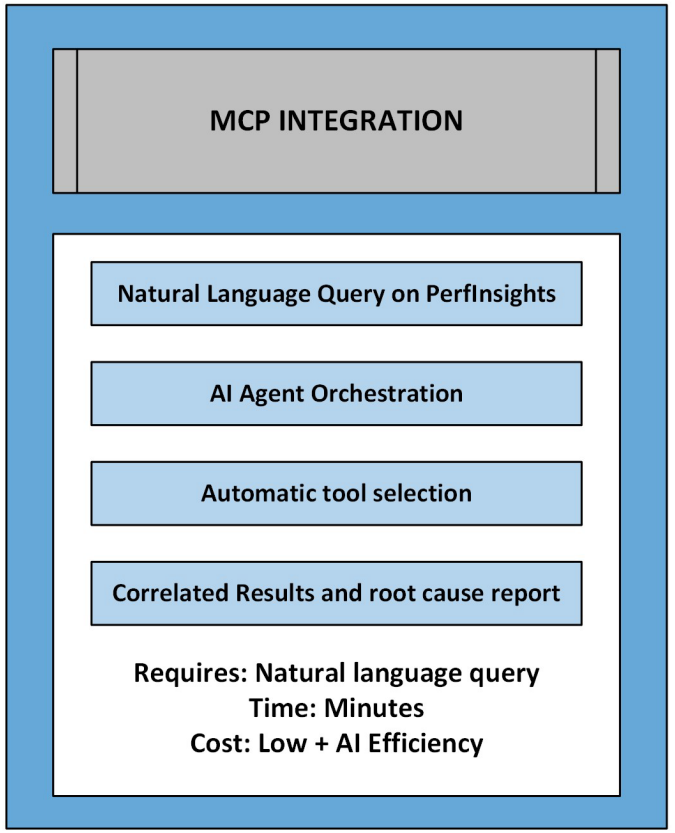
Cross-Domain Intelligence

- **Architectural Specs** - Expected BW, latency targets, buffer requirements
- **Trace Data** - Actual measurements, timestamps, packet journeys
- **Bug History** - Similar issues, known patterns, prior fixes

What Changes

- **Expertise needed** - Ask the question, get the diagnosis
- "Why is BW 15% low?" → AI selects tools, sequences analysis, correlates findings
- **Time** – Minutes | **Correlation** - Automatic

MCP Integration: AI Drives (Minutes)



Key Insight: *"Debug accelerates at the speed of conversation — natural language in, root cause out"*

The AI-Enhanced Debug Architecture

Layer 1 — Embedded Trackers

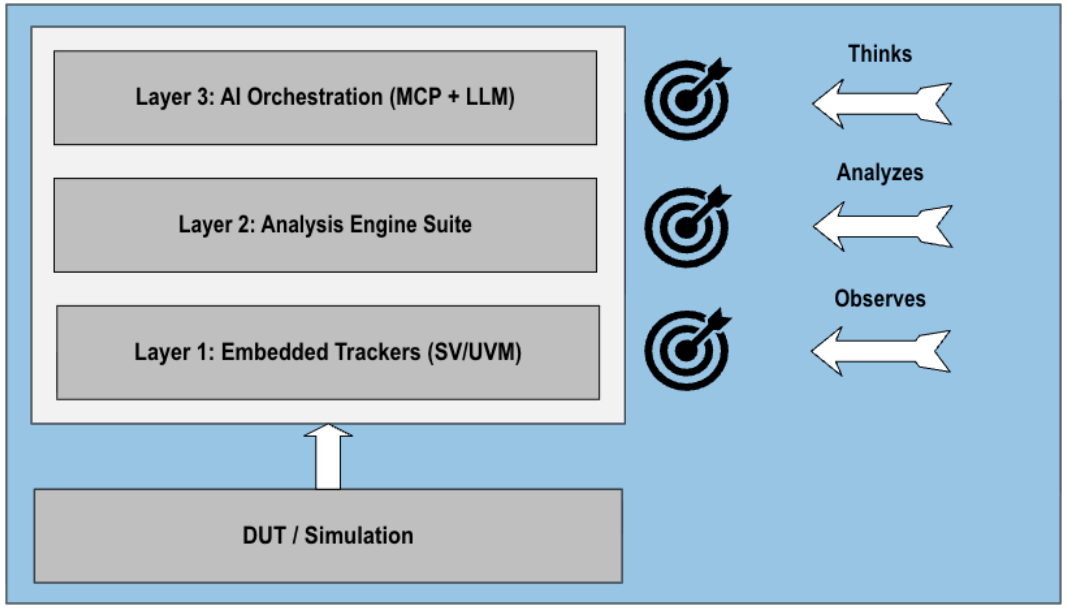
- Lightweight SystemVerilog monitors inside the DUT
- Real-time trace streaming during simulation

Layer 2 — Analysis Engine Suite

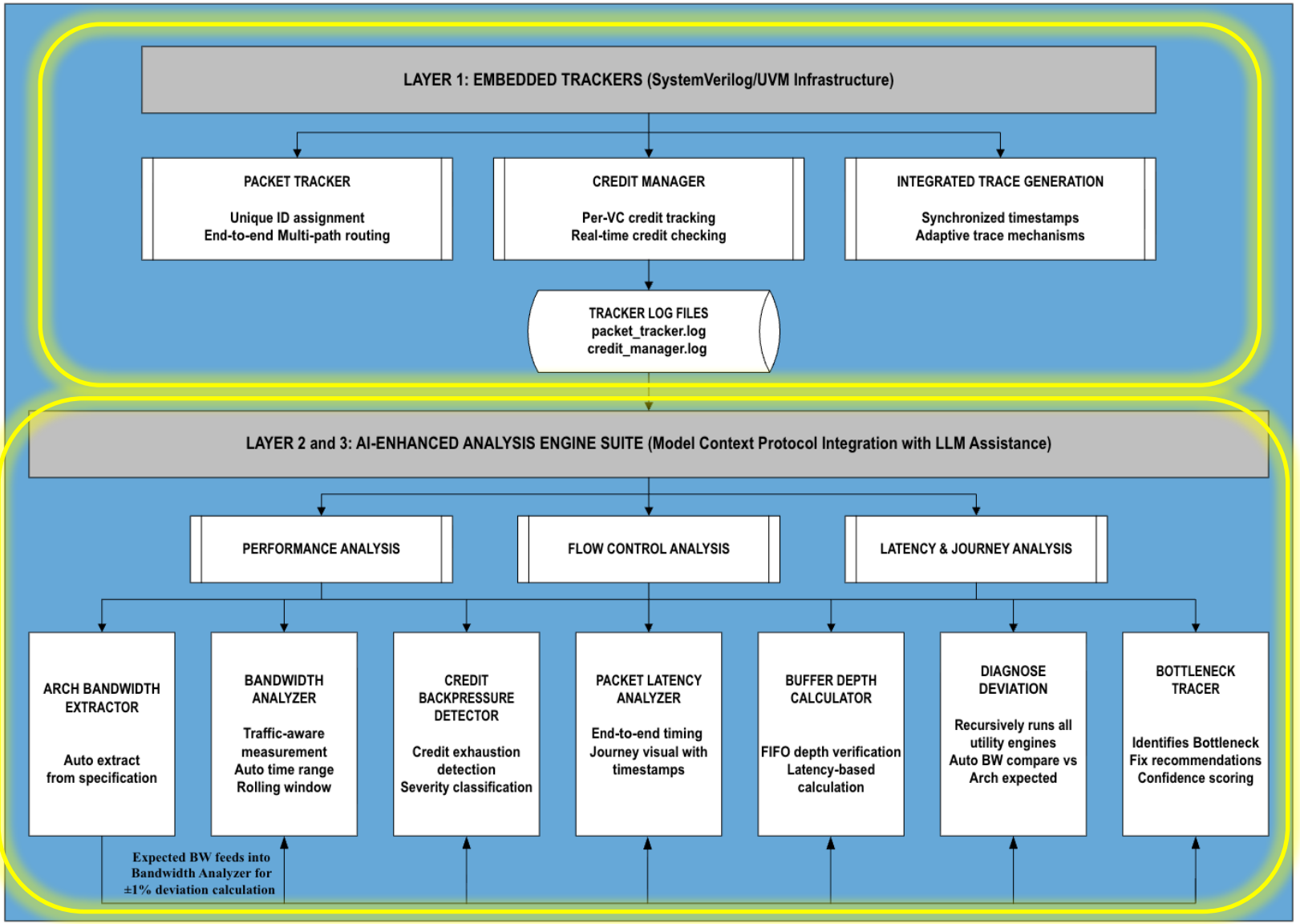
- Bandwidth analyzer with architecture spec integration
- Credit backpressure detector with severity classification
- Packet latency analyzer with pipeline visualization

Layer 3 — AI Integration (MCP)

- Natural language interface to entire debug workflow
- Automatic correlation across analysis engines
- Repeated tool(s) invocation until root cause is identified



Key Insight: *"Each layer amplifies the next — trackers feed engines, engines inform AI, AI orchestrates everything."*



Multi Layer Intelligence - Verification Environment with AI-Enhanced Debug Integration

Key Insight: *"Trackers capture. Engines analyze. AI orchestrates."*

Tracker Infrastructure - Packet

Lightweight, Real-Time Trace Generation – Packet

- **Embedded in DUT** - passive monitors, zero simulation overhead
- **Structured logs** - human-readable, **machine-parseable**
- **Replaces FSDB** - 10–100 MB vs. 50–200 GB per test

What It Tracks (End-to-end)

- **Full packet journey** - from TX injection to RX completion
- **Request ↔ Response correlation** - automatic round-trip pairing
- **Multi-GPU routing paths** - source, destination, intermediate hops

Why This Matters

- **Latency is measurable** - not estimated from waveform eyeballing
- **Bottlenecks are localized** - which stage? which GPU? which VC?
- **100% transactions captured** - no sampling, no gaps, no guessing

Field Reference

Field	Description
[T_xxxx]	Unique packet ID — tracks full journey
@ time	Timestamp (ps/ns granularity)
TX/RX	Transmit/Receive boundary crossing
CMD	WRITE, READ, RSP_RD, RSP_WR
SRC/DST	Source → Destination GPU
VC	Virtual Channel (REQ/RSP)

Trace Logs vs. FSDB

	Trace Logs	FSDB
Size	10–100 MB	500 GB–2 TB
Generation	**Real-time**	2–4 weeks
Coverage	100% tests	~10% tests
Time-to-Insights	Minutes	Days
Regression-Ready	Always-on	On-demand only
Automation	Fully scriptable	EDA tools

Key Insight: *"Architectural events tell you what matters — not every bit transition"*

Tracker Infrastructure - Packet

Tag	Timestamp (ps)	Direction	Command	Source	Destination	Details
T_0001	1000000000	TX_REQ	WRITE	GPU_A	GPU_C	Size_X
T_0002	1000025600	TX_REQ	WRITE	GPU_A	GPU_C	Size_X
T_0003	1000051200	TX_REQ	READ	GPU_B	GPU_D	Size_Y
T_0004	1000076800	TX_REQ	WRITE	GPU_D	GPU_B	Size_X
T_0001	1000085400	RX_REQ	WRITE	GPU_A	GPU_C	VC=REQ
T_0005	1000102400	TX_REQ	READ	GPU_C	GPU_A	Size_Z
T_0002	1000111000	RX_REQ	WRITE	GPU_A	GPU_C	VC=REQ
T_0003	1000136600	RX_REQ	READ	GPU_B	GPU_D	VC=REQ
T_0001	1000142300	TX_RSP	RSP_WR	GPU_C	GPU_A	ACK
T_0004	1000162200	RX_REQ	WRITE	GPU_D	GPU_B	VC=REQ
T_0003	1000214400	TX_RSP	RSP_RD	GPU_D	GPU_B	Size_Y
T_0005	1000187800	RX_REQ	READ	GPU_C	GPU_A	VC=REQ
T_0001	1000227700	RX_RSP	RSP_WR	GPU_C	GPU_A	ACK
T_0002	1000268000	TX_RSP	RSP_WR	GPU_C	GPU_A	ACK
T_0003	1000299800	RX_RSP	RSP_RD	GPU_D	GPU_B	Size_Y
T_0005	1000265600	TX_RSP	RSP_RD	GPU_A	GPU_C	Size_Z
T_0004	1000319100	TX_RSP	RSP_WR	GPU_B	GPU_D	ACK
T_0002	1000353400	RX_RSP	RSP_WR	GPU_C	GPU_A	ACK
T_0005	1000351000	RX_RSP	RSP_RD	GPU_A	GPU_C	Size_Z
T_0004	1000404500	RX_RSP	RSP_WR	GPU_B	GPU_D	ACK

Key Insight: *"Structured logs turn debug into data science"*

Tracker Infrastructure - Credit

Lightweight, Real-Time Trace Generation - Credit

- Monitor credit-based flow control across all virtual channels (VCs)
- Detect backpressure, starvation, and credit exhaustion in real-time

Why This Matters

- Bandwidth depends on credits - no credits = no packets sent
- Invisible in FSDBs: credit state requires deep signal correlation

The Transformation

- From *"BW dropped"* → to where, when, and why
- From *"something blocked"* → to which VC, how long, how severe
- From *"check the waveform"* → to root cause delivered

What It Tracks (Per VC, Per Interface)

Event	Description
CDT_INC	Credit returned (buffer freed downstream)
CDT_DEC	Credit consumed (packet sent)
AVAILABLE	Current credit count per VC
STATUS	NORMAL → LOW → STARVED → EXHAUSTED

Severity Classification (Configurable)

Status	Condition	Impact
NORMAL	credits ≥ 8	Full throughput
LOW	0 < credits < 8	Throughput risk
STARVED	credits = 0	Traffic stalled
EXHAUSTED	credits = 0 for >100ns	Sustained blockage

Key Insight: *"Credits tell the truth that bandwidth hides"*

Tracker Infrastructure - Credit

Timestamp (ps)	VC	Action	Current Credits	Status
1,000,000,000	REQ_VC0	CDT_DEC	62	NORMAL
1,000,025,600	REQ_VC0	CDT_DEC	60	NORMAL
1,000,051,200	REQ_VC0	CDT_DEC	58	NORMAL
1,000,076,800	RSP_VC0	CDT_INC	64	NORMAL
1,000,102,400	REQ_VC0	CDT_DEC	32	NORMAL
1,000,128,000	REQ_VC0	CDT_DEC	16	NORMAL
1,000,153,600	REQ_VC0	CDT_DEC	6	LOW ⚠
1,000,179,200	REQ_VC0	CDT_DEC	2	LOW ⚠
1,000,204,800	REQ_VC0	CDT_DEC	0	STARVED 🔴
1,000,230,400	REQ_VC0	-------	0	STARVED 🔴
1,000,256,000	REQ_VC0	CDT_INC	4	LOW ⚠
1,000,281,600	REQ_VC0	CDT_INC	16	NORMAL
1,000,307,200	REQ_VC0	CDT_INC	48	NORMAL

Key Insight: *"Traffic stalls where credits starve — now you see exactly where"*

Analysis Engine Suite

The Toolchain: From Traces to Root Cause

- **8 specialized analyzers** — each solves a specific performance question
- **Composite orchestrators** — Iterates tool invocation
- **MCP-enabled** — AI agents invoke tools through natural language

How They Work Together

- Trace Logs → Invokes Individual Analyzers → Cross-Correlation → Unified Diagnosis → Root cause + Fix recommendation

BW Engines	What It Does
Arch BW Extractor	Pulls expected bandwidth from architecture specs (Excel/HDL)
Bandwidth Analyzer	Measures actual BW from traces, compares to expected (±1% tolerance)

"Is my test hitting spec?" → Deviation %, severity, rolling window trends

Flow Control Engines	What It Does
Credit Backpressure Detector	Finds sustained low-credit / starvation periods
Severity Classification	NORMAL → LOW → STARVED → EXHAUSTED

"Why did bandwidth drop?" → Credit exhaustion at T=X on VC=Y for Z ns

Key Insight: *"Bandwidth, credits, latency — triangulate the truth: Orchestration diagnoses"*

AI Development Methodology

The Four-Stage Process

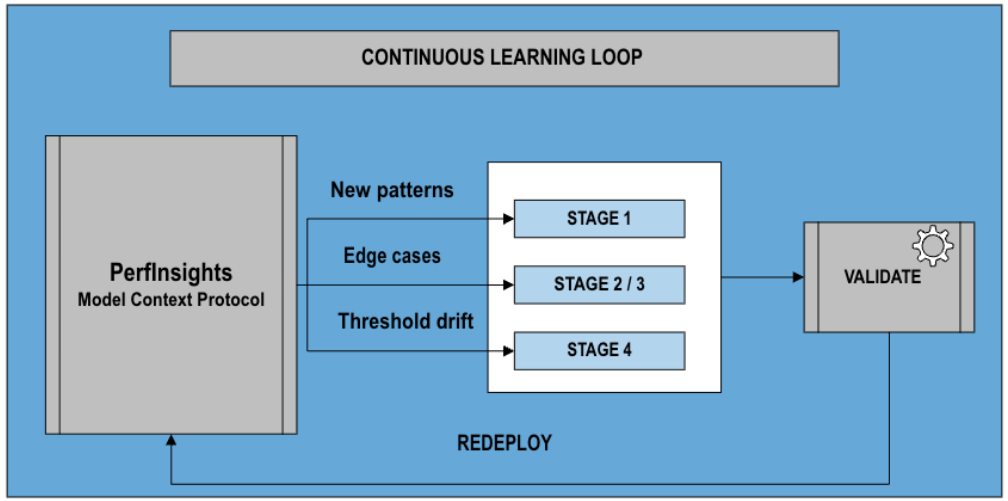
Expert Knowledge → Prototype → AI Enhancement → Validation → Production

Stage 1: Pattern Extraction → *Expertise captured*

Stage 2: Algorithm Prototyping → *Logic encoded*

Stage 3: LLM Code Generation → *Development accelerated*

Stage 4: Cross-Validation → *Accuracy proven*



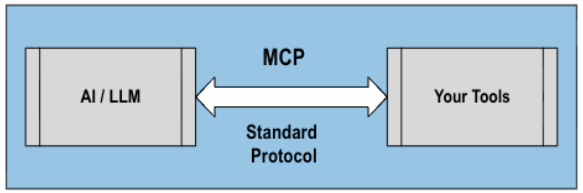
Continuous Learning Loop

- **New patterns discovered** → feed back to Stage 1
- **Edge cases found** → refine algorithms in Stage 2/3
- **Threshold drift** → recalibrate in Stage 4

The tools get smarter with every debug session.

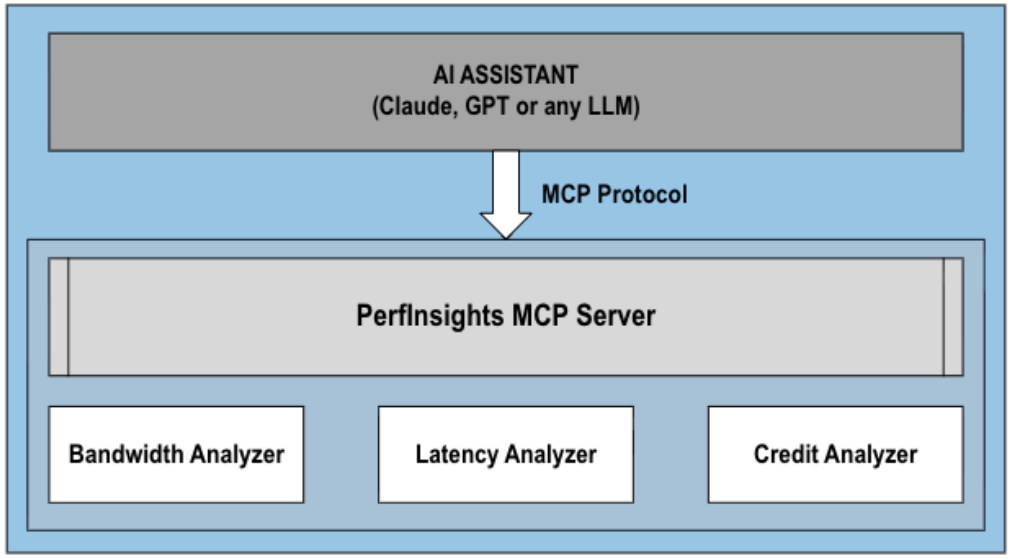
Key Insight: *"Human expertise seeds the algorithm. AI scales it. Validation proves it."*

PerfInsights MCP



PerfInsights: Our MCP Server

- **Exposes all analysis engines** through standardized MCP interface
- **Natural language accessible**
- **Zero learning curve** — describe the problem, get the diagnosis



Why MCP Matters for Verification

Before MCP	With MCP
Custom integrations per tool	One protocol, all tools
Brittle, hard-coded pipelines	Dynamic tool discovery
AI can't "see" your infra	Access to logs, specs, analyzers
Manual copy-paste to AI	Direct, structured interaction

Available Analysis Engines via MCP

Tool	Natural Language Trigger
analyze_bandwidth	"What's the actual BW?"
extract_perf_stats	"What's the expected BW?"
analyze_credit_backpressure	"Any credit issues?"
analyze_packet_latency	"Where's latency coming from?"
detect_backpressure	"Is there backpressure?"
validate_init_credits	"Are credits configured correctly?"
diagnose_link	"Debug link X"
find_bottleneck	"What's the limiting factor?"

Key Insight: *"MCP turns tools into teammates - Talk to your tools, they solve your problems."*

PerfInsights MCP - AI-Orchestrated Workflow

The Query: *"Why is my test showing 15% bandwidth loss?"*

Key Capabilities

- **Dynamic tool selection** — not a fixed script, adapts to findings
- **Early exit optimization** — stops if root cause found
- **Cross-domain integration** — arch specs + traces + bug history
- **Severity classification** — prioritizes what matters

AI Orchestration Sequence

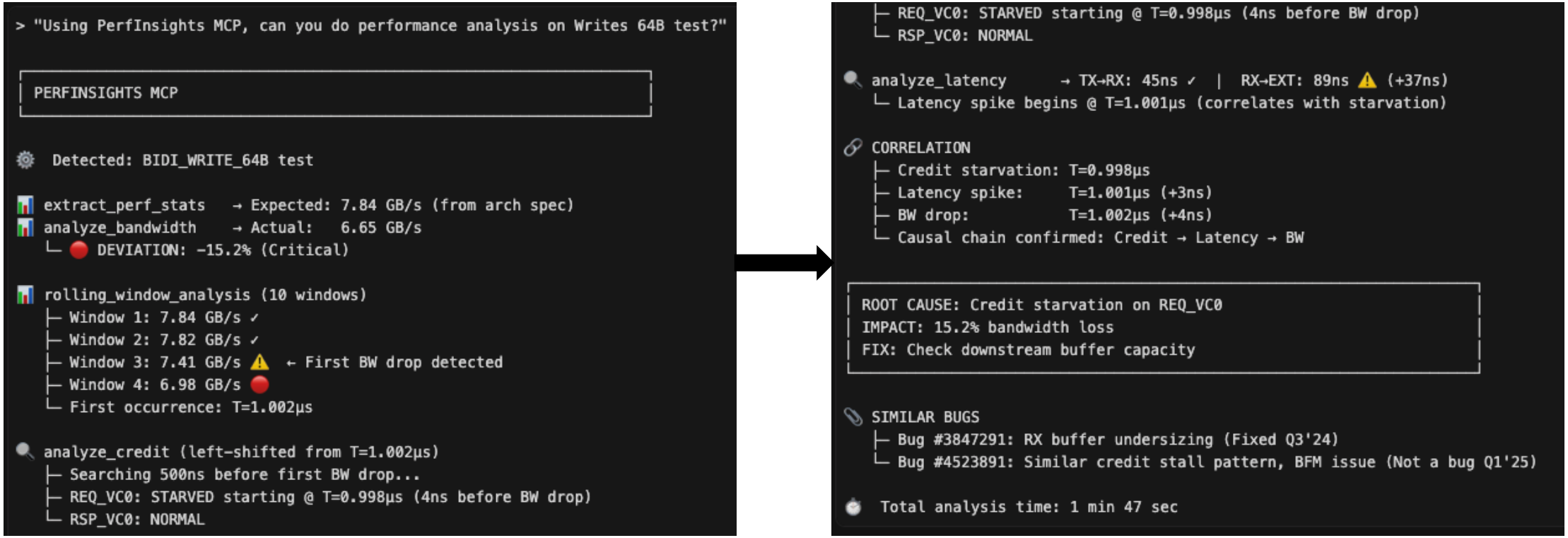
Step	Action	Tool Invoked
1	Parse test directory, detect config	Auto-detection
2	Extract expected BW from arch specs	extract_perf_stats
3	Measure actual BW from traces	analyze_bandwidth
4	Calculate deviation (15.2% below spec)	Comparison engine
5	Check for credit issues	analyze_credit_backpressure
6	Profile latency breakdown	analyze_packet_latency
7	Cross-correlate all findings	Correlation engine
8	Deliver root cause + recommendation	Final synthesis

What AI Decides Automatically

Decision	How AI Chooses (MCP)
Which tools to run	Based on symptom keywords + test configuration
In what order	Dependency-aware sequencing
What to skip	If credits are normal, skip deeper credit analysis
How to correlate	Timestamp alignment, causal reasoning
What to recommend	Pattern matching to known fixes

Key Insight: *"AI doesn't just run tools — it thinks through the debug"*

PerfInsights MCP – Visual Flow



Key Insight: *"The full debug — automated, correlated, referenced"*

Case Study #1- Buffer Undersizing in Datapath

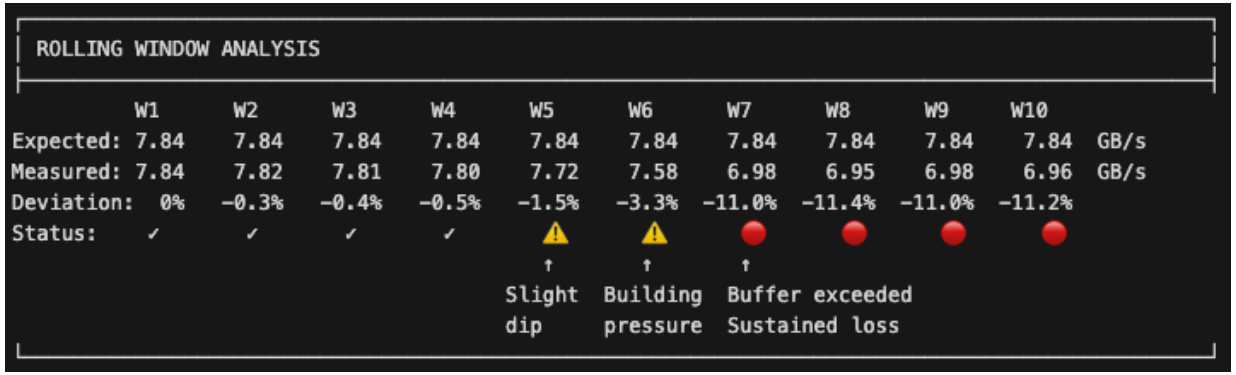
The Symptom: > **11% bandwidth degradation** in bidirectional write traffic

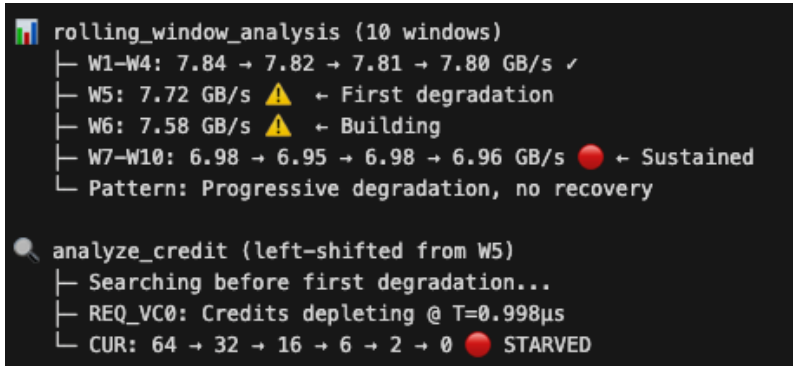
> *Use PerfInsights to debug write traffic performance issue.*

AI-Orchestrated Diagnosis

Step	Tool	Finding
1	Bandwidth Analyzer	BW dropped from 7.84 → 6.98 GB/s
2	Rolling Window Analysis	Progressive degradation — pinpointed at Window 5
3	Arch BW Extractor	Confirmed 11% deviation vs Excel spec
4	Credit Backpressure Detector	RX-side credit exhaustion in same window
5	Buffer Depth Calculator	Undersizing flagged vs BW-latency requirements

The Correlation





Key Insight: *"Rolling windows caught what averages hid"*

Case Study #1- Buffer Undersizing in Datapath

Root Cause

- **Receive buffer undersized** - insufficient depth for bandwidth-delay product
- Rolling window analysis pinpointed **exact packet range** where credit backpressure began

The Fix

- ✓ **Increased buffer depth** per architectural guidelines

What Made the Difference

- **Rolling window** - caught progressive degradation, not just average
- **Cross-tool correlation** - BW drop + credit starvation + buffer sizing aligned
- **Automatic spec comparison** - deviation quantified instantly
- **Architectural validation** - buffer depth checked against requirements

Time Comparison

Criteria	Manual Debug	AI-Orchestrated
Time	3–4 days	8 minutes
Tools used	Unknown sequence	5 tools, optimal order
Root cause clarity	Eventually found	Immediate

Key Insight: *"Three days of hunting → eight minutes of asking"*

Case Study #2- Credit Return Bug with Mixed Traffic

The Symptom:

- > **7% bandwidth loss** with 50% read / 50% write mixed traffic (read/write commands)

The Puzzle

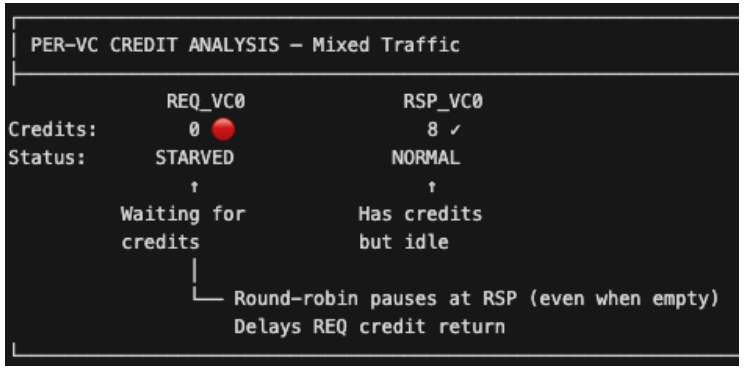
Traffic Pattern	Measured BW	Status
Write-only	7.9 GB/s	✅ Pass
Read-only	7.9 GB/s	✅ Pass
Mixed 50/50	7.3 GB/s	❌ 7% loss

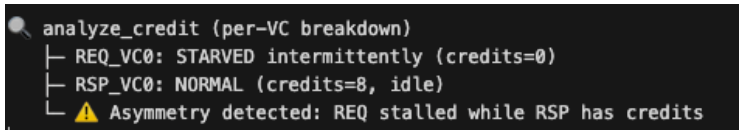
AI-Orchestrated Diagnosis

> Why does mixing traffic break bandwidth?

Step	Tool	Finding
1	Bandwidth Analyzer	Detected 7% degradation in mixed mode only
2	Per-VC Breakdown	REQ VC stalled while RSP VC had credits available
3	Credit Flow Analysis	Credit return delay correlated with mixed command sizes
4	Pattern Matching	Round-robin arbitration inefficiency identified

The Correlation





Key Insight: *"Aggregate metrics pass. Per-VC metrics caught the bug"*

Case Study #2- Credit Return Bug with Mixed Traffic

Root Cause and the fix:

- Round-robin credit return bug: Skip VCs with zero credits immediately — no extra cycle pause

Why This Bug Was Hard to Find Manually

- **Pass in isolation** — write-only ✓, read-only ✓
- **Subtle timing** — one extra cycle, buried in thousands.

Fail only in combination — mixed traffic corner case

Per-VC visibility required — aggregate BW hides the cause

What Made the Difference

- **Per-VC credit breakdown** — saw REQ stalled while RSP idle
- **Traffic-aware analysis** — compared modes automatically
- **Pattern isolation** — linked to command size variation
- **No waveforms needed** — trace data was sufficient

Time Comparison

Attribute	Manual Debug	AI-Orchestrated
Time	1 week	10 minutes
Waveforms analyzed	Thousands of cycles	Zero
Root cause clarity	Hard to isolate	Per-VC breakdown exposed it

Key Insight: *"One extra cycle. One week of debug. Ten minutes with AI"*

Case Study #3- BFM Response Buffer Modeling Issue

The Symptom

> **3% bandwidth loss** in weighted read/write traffic modeling multi-GPU interactions. Expected: 7.8 GB/s | Actual: 7.6 GB/s

The Misleading Data

Metric	Value	Status
Aggregate read/write ratio	50/50	✅ Correct
Total packet count	Matched	✅ Correct
Overall BW	7.6 GB/s	❌ 3% low

AI-Orchestrated Diagnosis

> Aggregate statistics said everything was fine. But BW was still low.

Step	Tool	Finding
1	Single-Window Analysis	Aggregate 50/50 mix — looks correct
2	Rolling Window Analysis	Detected temporal imbalances
3	Per-Window Breakdown	Window 4: 55% reads, Window 7: 45% writes, Window 9: 55% reads
4	Credit Return Correlation	Timing dependencies traced to response buffer

Key Insight: *"Aggregates lie. Rolling windows tell the truth"*

Case Study #3- BFM Response Buffer Modeling Issue

Root Cause

- **BFM response buffer held responses longer than specified** > Issue was in **test infrastructure**

The Fix

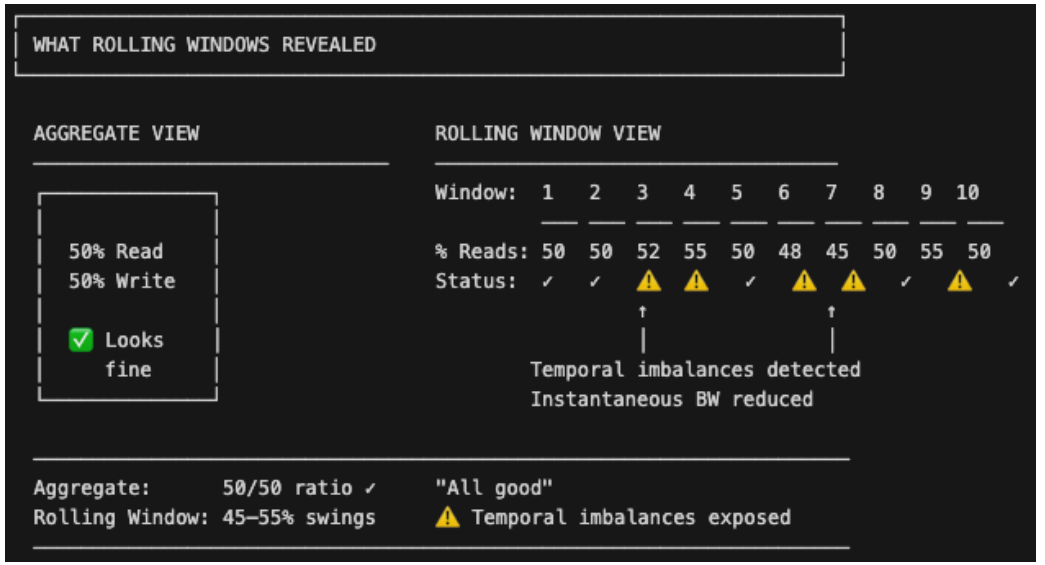
- ✓ **Modified BFM response buffer** to follow architecture-specified memory latency
- ✓ Temporal imbalances eliminated ✓ BW restored to 7.8 GB/s

Why This Was Dangerous

- **Aggregate metrics passed** — 50/50 ratio looked correct
- **HW was blamed first** — weeks could be wasted debugging good RTL
- **Subtle temporal effect** — only visible in per-window breakdown

What Made the Difference

- **Prevented false HW debug** — saved weeks of wasted effort



Time Comparison

	Manual Debug	AI-Orchestrated
Time	1–2 weeks	12 minutes
False HW debug time	Days wasted	Zero
Root cause location	HW suspected first	BFM identified directly

Key Insight: *"Temporal analysis prevented weeks of debugging the wrong thing"*

Results - The Transformation

Debug Efficiency

Metric	Traditional FSDB	AI-Enhanced Trace	Impact
Root Cause Identification	2–4 weeks	1–4 hours	50–80× faster
Engineer Hours per Issue	40–80 hrs (Arch+RTL+DV)	1–2 hours (AI-orchestrated)	40–80× reduction
Cross-Team Coordination	Manual (3+ teams)	AI autonomous	Eliminated
First-Try Accuracy	60% (manual)	97–98% (AI-guided)	60% better
False Positive Rate	15%	1–3%	6× better

Scalability

Metric	Traditional FSDB	AI-Enhanced Trace	Impact
Full-Chip Tests (100s GB sims)	Infeasible	3–10 min analysis	Enabled
Regression Throughput	50 tests/day	5000 tests/day	100×

Key Insight: *"Two orders of magnitude faster. Ten times the coverage. Zero waveforms"*

Results - The Bottom Line

AI-Driven Innovation

Capability	Traditional	AI-Enhanced	Impact
Natural Language Debug	✗	✓ (MCP-enabled)	New paradigm
Multi-Tool Orchestration	✗ (manual)	✓ Autonomous	Breakthrough
Architectural Spec Integration	✗	✓ Auto-query IAS	First in EDA
Bug Repository Correlation	✗	✓ Semantic search	Novel

The Bottom Line

Before	After
10% of tests analyzed	100% of tests analyzed
Weeks to root cause	Minutes to root cause
3+ teams coordinating	AI orchestrates autonomously
TB of storage	GB of storage
FSDB generation delays	Real-time traces

Key Insight: *"80× speedup. 10× coverage. 100× throughput"*

Future Vision

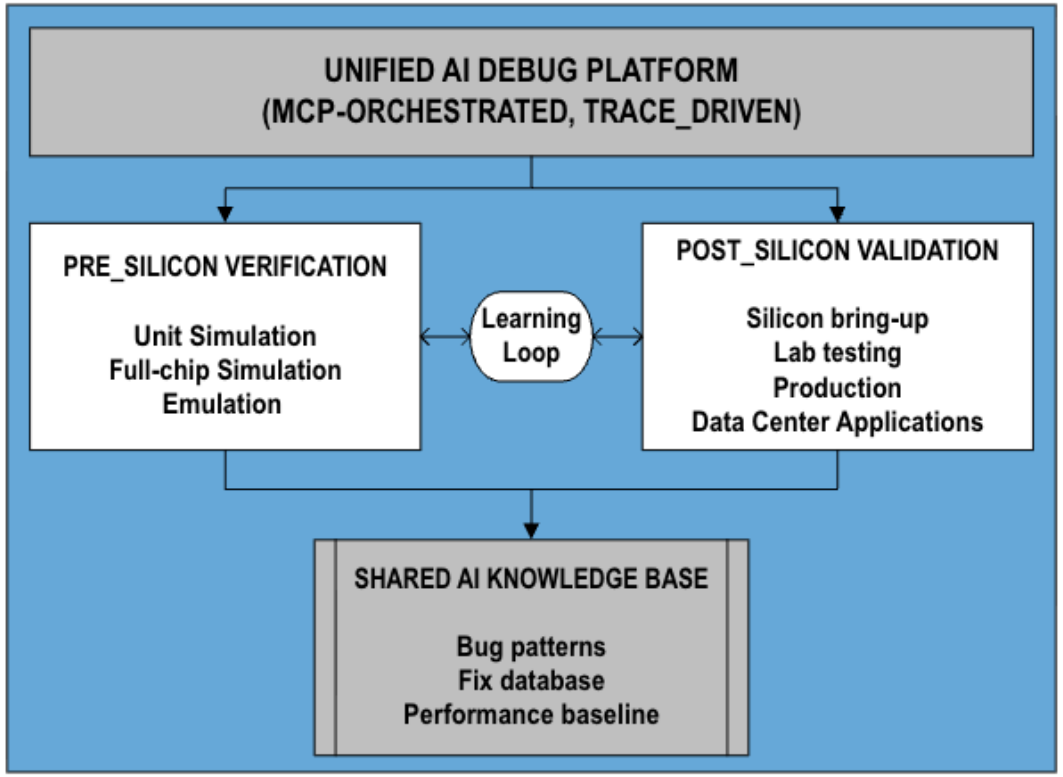
The Gap Today

- **Verification teams** debug simulation failures in isolation
- **Validation engineers** analyze production hardware separately
- **Different toolchains** — no knowledge transfer between domains

The Insight

- **Trace-driven analysis is inherently cross-domain compatible**

Pre-Silicon	Post-Silicon	Same Semantics
SystemVerilog trackers	On-chip performance monitors	✓
Packet/credit logs	JTAG interfaces, PCIe telemetry	✓
Timestamps, BW, latency	Timestamps, BW, latency	✓



Key Insight: *"Pre-silicon patterns predict post-silicon failures. Now AI connects them"*

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

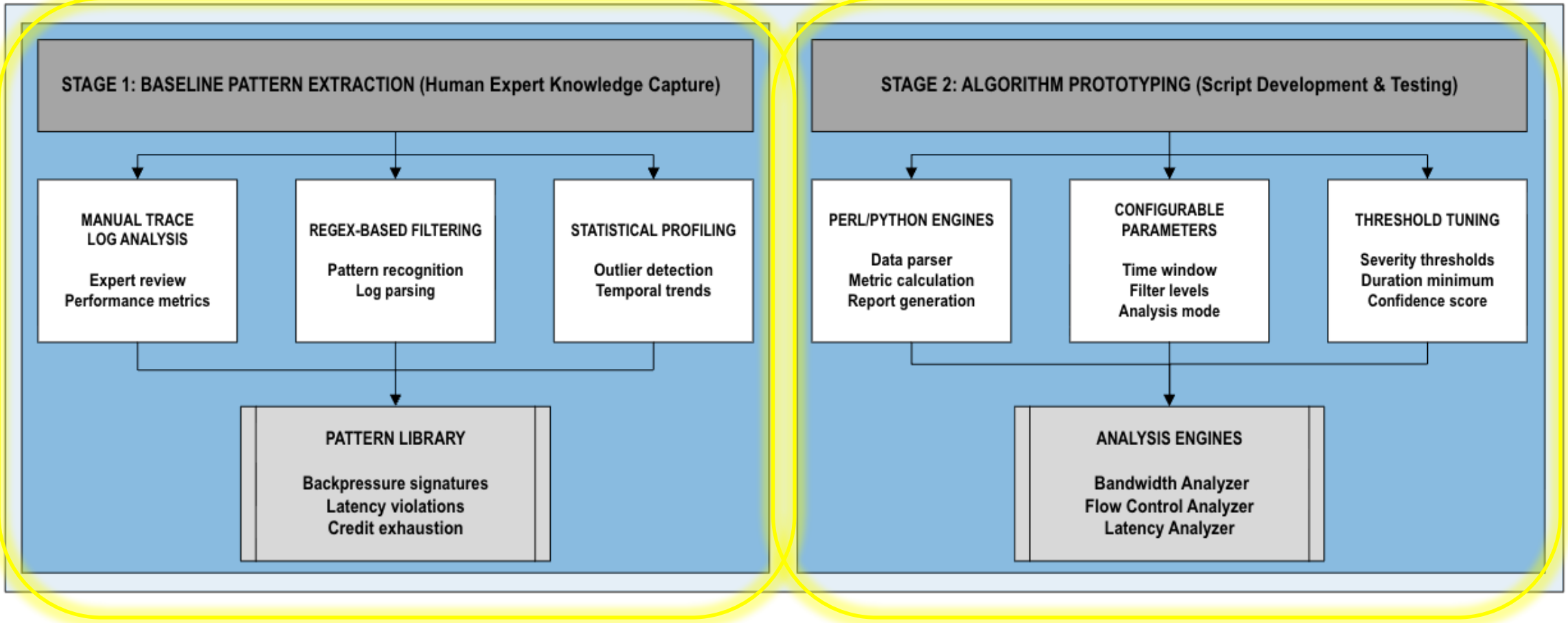
Thank You !

Q & A

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

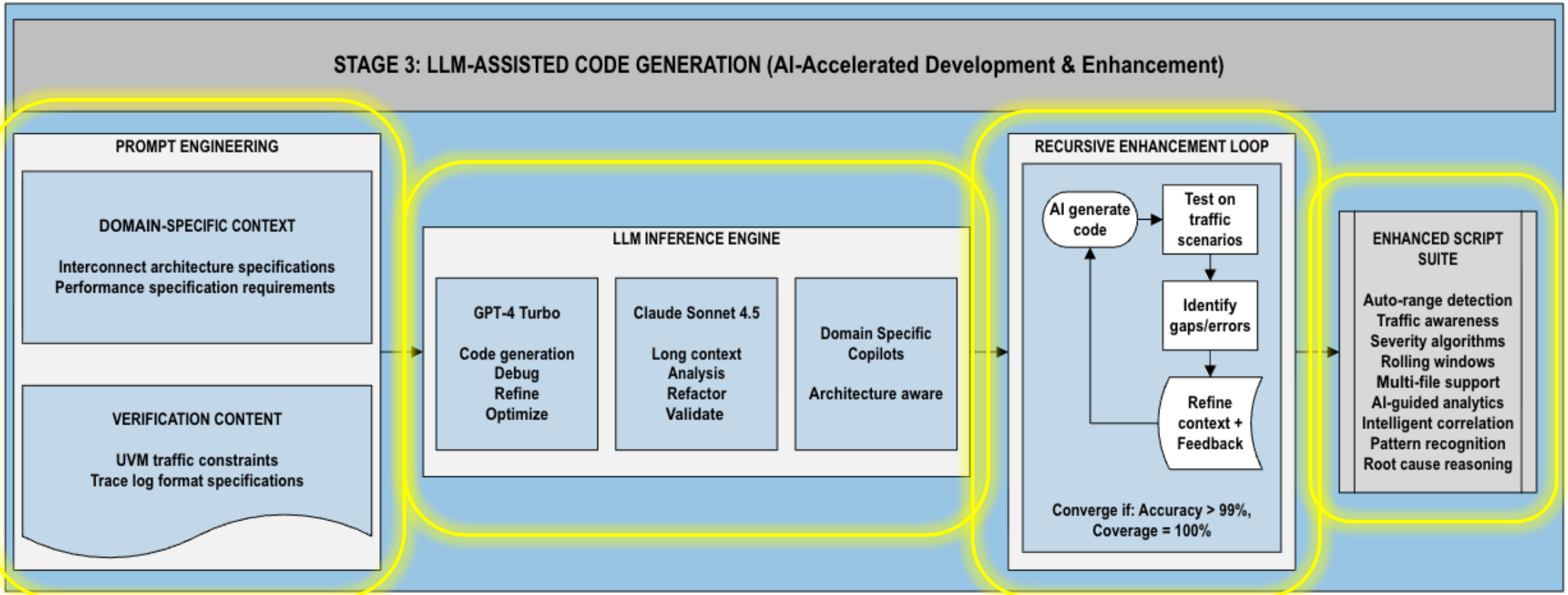
Backup Slides

AI Development Methodology



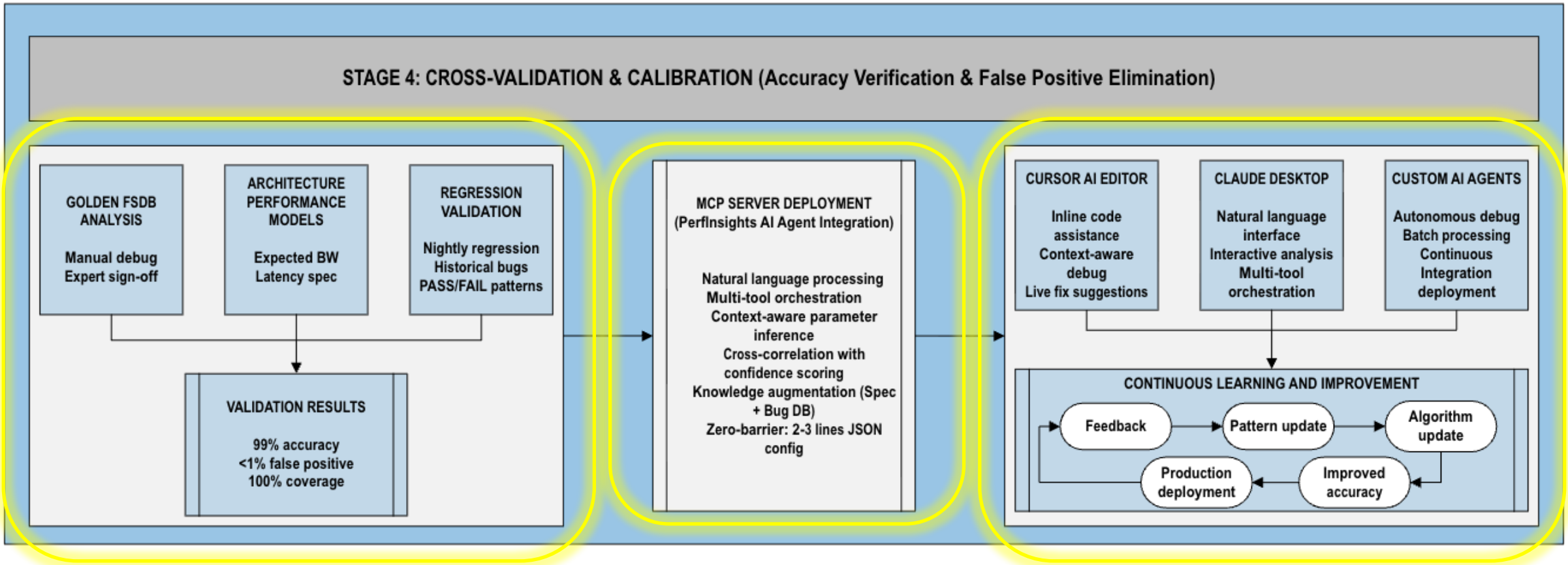
Key Insight: *"Capture patterns. Code logic. Tune thresholds - Encode once what experts learned over years"*

AI Development Methodology



Key Insight: *"Prompt → Generate → Test → Refine → Converge : Engineers guide. LLMs generate. Tests validate"*

AI Development Methodology



Key Insight: *"Every deployment makes the next one smarter - Accuracy is earned, not assumed"*

The AI-Enhanced Debug Architecture

Layer 1 — Embedded Trackers

- Lightweight SystemVerilog monitors inside the DUT
- Real-time trace streaming during simulation

Layer 2 — Analysis Engine Suite

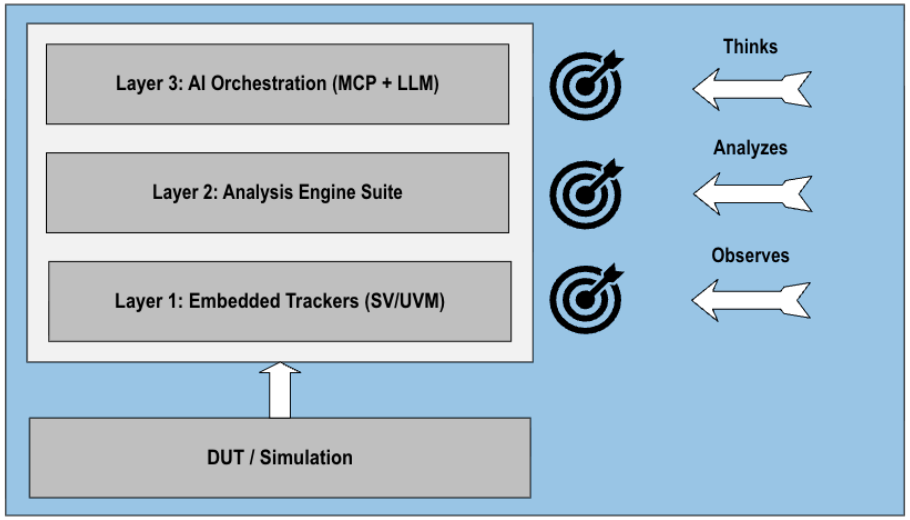
- Bandwidth analyzer with architecture spec integration
- Credit backpressure detector with severity classification
- Packet latency analyzer with pipeline visualization

Layer 3 — AI Integration (MCP)

- Natural language interface to entire debug workflow
- Automatic correlation across analysis engines
- Repeated tool(s) invocation until root cause is identified

The Three Layers

Layer	What It Does	Key Benefit
Embedded Trackers	Lightweight SystemVerilog monitors stream traces in real-time	Zero overhead, human-readable logs
Analysis Engines	Specialized tools for bandwidth, credits, latency, buffers	Domain expertise encoded as automation
AI Orchestration	MCP enables LLMs to coordinate multi-tool workflows	Natural language → Root cause



Key Insight: *"Each layer amplifies the next — trackers feed engines, engines inform AI, AI orchestrates everything."*

Analysis Engine Suite

The Toolchain: From Traces to Root Cause

- **8 specialized analyzers** — each solves a specific performance question
- **Composite orchestrators** — Iterates tool invocation
- **MCP-enabled** - AI agents invoke tools through natural language

BW Engines	What It Does
Arch BW Extractor	Pulls expected bandwidth from architecture specs (Excel/HDL)
Bandwidth Analyzer	Measures actual BW from traces, compares to expected (±1% tolerance)

"Is my test hitting spec?" → Deviation %, severity, rolling window trends

Flow Control Engines	What It Does
Credit Backpressure Detector	Finds sustained low-credit / starvation periods
Severity Classification	NORMAL → LOW → STARVED → EXHAUSTED

"Why did bandwidth drop?" → Credit exhaustion at T=X on VC=Y for Z ns

How They Work Together

- Trace Logs → Individual Analyzers → Cross-Correlation → Unified Diagnosis → Root cause + Fix recommendation

Engine	What It Does
Packet Latency Analyzer	Measures end-to-end packet timing and provides a detailed block-by-block latency breakdown to pinpoint where delays occur.
Buffer Depth Calculator	Verifies FIFO buffer sizing by checking against the bandwidth-delay product, ensuring buffers are sufficiently dimensioned for expected traffic.
Diagnose Deviation	Executes all diagnostic engines, correlates their outputs, and suggests potential root causes for observed performance deviations.
Bottleneck Tracer	Analyzes system performance data to rank issues by severity and identifies the primary limiting factor affecting throughput or latency.

"Where's my latency coming from?" → Pipeline stage X added 47ns (vs:12ns)

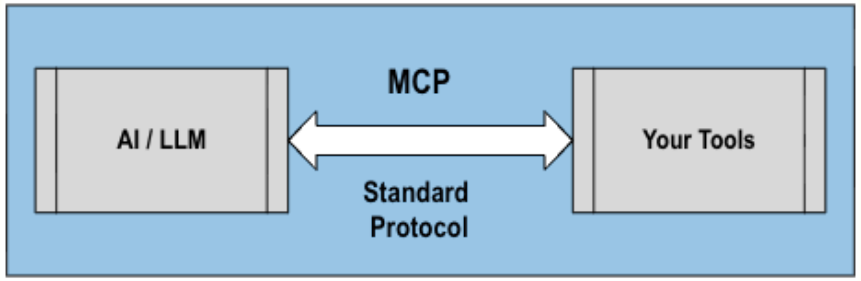
Key Insight: *"Bandwidth, credits, latency — triangulate the truth: Orchestration diagnoses"*

Model Context Protocol MCP

What is MCP?

An open protocol that standardizes how AI assistants (LLMs) connect to external tools, data sources, and systems.

- **Model Context Protocol** — an open standard for AI ↔ Tool communication
- **Think "USB for AI"** — universal connector between LLMs and external tools
- **Standardized interface** — any AI agent can invoke any MCP-enabled tool
- **Bidirectional** — AI sends queries, tools return structured results



Why MCP Matters for Verification

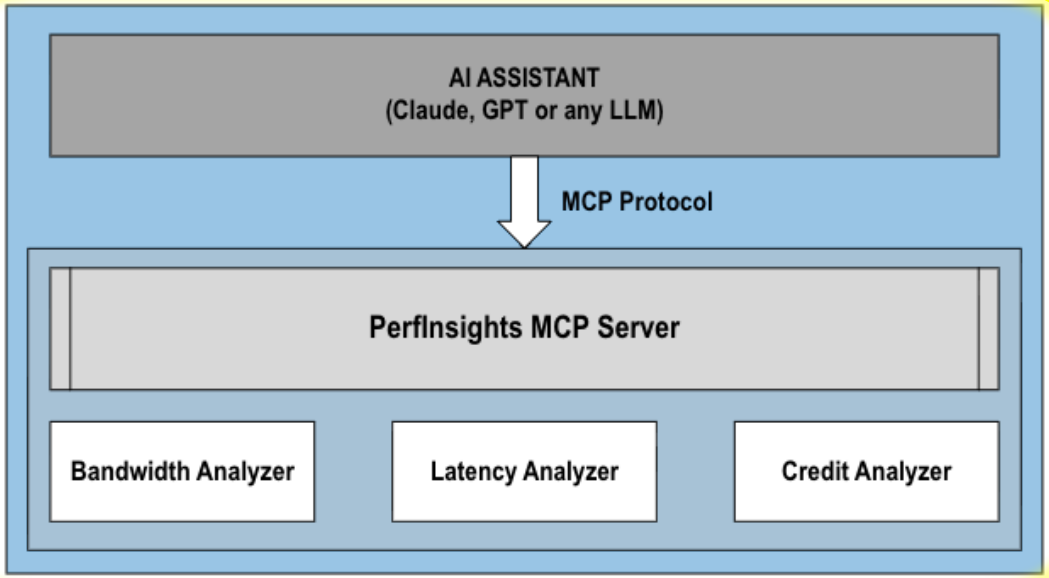
Before MCP	With MCP
Custom integrations per tool	One protocol, all tools
Brittle, hard-coded pipelines	Dynamic tool discovery
AI can't "see" your infrastructure	AI accesses logs, specs, analyzers
Manual copy-paste to AI	Direct, structured interaction

Key Insight: *"MCP is the bridge between natural language and your toolchain."*

PerfInsights MCP

PerfInsights: Our MCP Server

- **Exposes 8 analysis engines** through standardized MCP interface
- **Natural language accessible** — Claude, Cursor AI, custom agents
- **Zero learning curve** — describe the problem, get the diagnosis



Available Tool via MCP

Tool	Natural Language Trigger
analyze_bandwidth	"What's the actual BW?"
extract_perf_stats	"What's the expected BW?"
analyze_credit_backpressure	"Any credit issues?"
analyze_packet_latency	"Where's latency coming from?"
detect_backpressure	"Is there backpressure?"
validate_init_credits	"Are credits configured correctly?"
diagnose_link	"Debug link X"
find_bottleneck	"What's the limiting factor?"

AI Agent Integration

Platform	How It Uses PerfInsights
Cursor AI	Inline debug in editor — ask questions, get answers
Claude Desktop	Interactive multi-tool analysis sessions
Custom Agents	Autonomous regression analysis, batch processing

Why MCP Matters

Without MCP	With MCP
Learn 8 tool syntaxes	Ask in English
Manual tool sequencing	AI orchestrates
You correlate results	AI correlates
Expertise required	Expertise embedded

Key Insight: *"MCP turns tools into teammates - Talk to your tools, they solve your problems."*

Results - The Transformation

Verification Coverage

Metric	Traditional FSDB	AI-Enhanced Trace	Impact
Tests Analyzed per Regression	10% (100/1000)	100% (1000/1000)	10× better
Coverage Limitation	FSDB infeasible	Real-time traces	Eliminated

Storage & Infrastructure

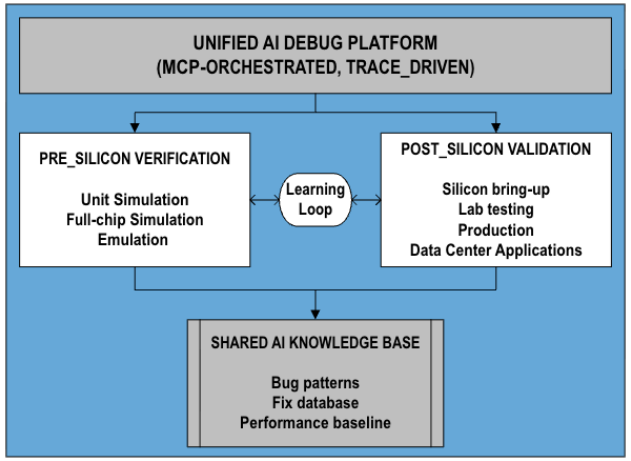
Metric	Traditional FSDB	AI-Enhanced Trace	Impact
Storage per Regression (1000 tests)	500 GB–2 TB	50–200 GB	10× better
FSDB Generation Time	2–4 weeks	0 (not required)	Eliminated
Waveform Tool Licenses	Expensive	$0	Eliminated

Key Insight: *"Two orders of magnitude faster. Ten times the coverage. Zero waveforms"*

Future Vision

The Unified Ecosystem

- **One MCP layer** — same orchestration, both domains
- **One knowledge base** — bugs, specs, fixes, baselines
- **One AI** — learns from both pre- and post-silicon
- **Format adapters** — LLM-assisted, auto-generated in hours (not months)



What This Enables

Capability	Impact
Cross-domain bug correlation	Post-silicon issue? AI checks if similar pattern exists in pre-silicon history
Retroactive learning	Production failure teaches AI to detect earlier in future designs
Unified performance baselines	Same metrics, simulation to data center
Elimination of domain boundaries	Insights flow freely across verification → validation

From Isolation to Orchestration

Today	Future
Separate tools per domain	Unified MCP platform
Manual knowledge transfer	AI-maintained knowledge base
Expertise in isolation	Shared insights
Months for tool porting	Hours for format adapters

Key Insight: *"One platform. Simulation to data center. Insights everywhere"*