2026 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

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

AI-Driven Adaptive Emulation for Accelerated Pre-Silicon Debug: High-Fidelity Silicon Replay

Sampathkumar M Ballary, Priyadarshini Pai, Aruna S Lohiya, Ramesh G Patgar, Karthik Srinivasan, Vemuri Krishna Sumanth, Vandana S, Venkata Subba Rao Manne, Subrata Sarkar, Samruddhi C R

Samsung Semiconductor India Research

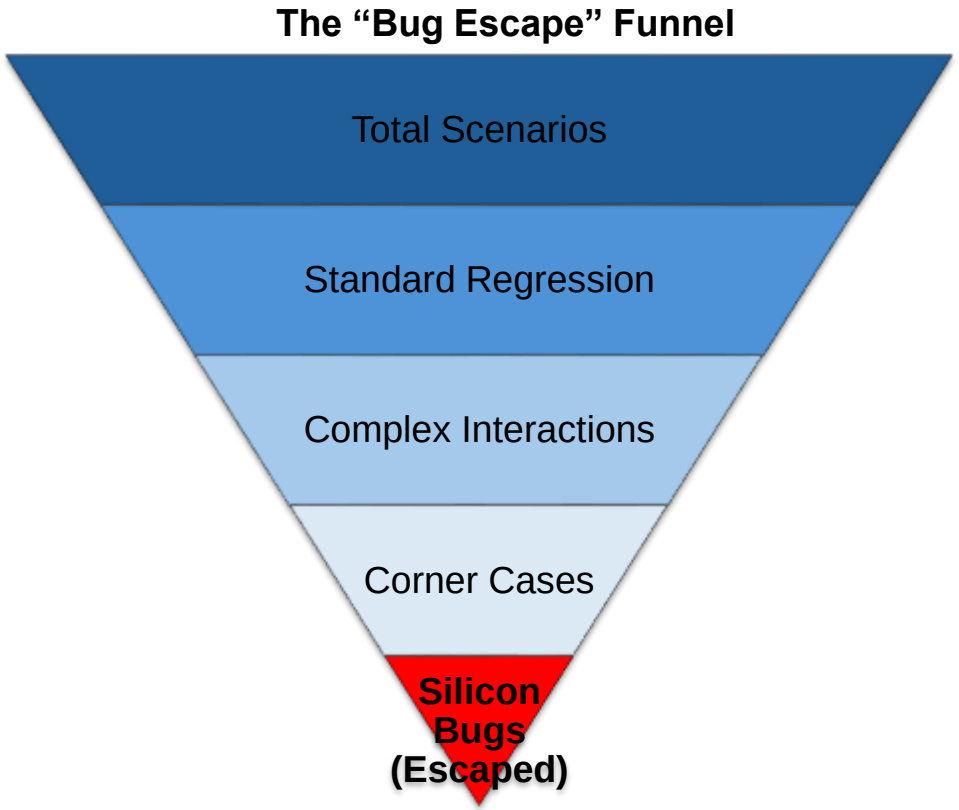
Agenda

- Prologue: The Debug Complexity Imperative
- Problem Statements
- Introduction to our Solution
- Adaptive Traffic Generator
- Adaptive Bus Modelling Engine
- Agentic AI based Constraints Solver & Vector Generator
- Results & Performance benchmarking v/s S.O.T. A
- Conclusion and Future Work

Prologue : The Debug Complexity Imperative

- Today’s AI-powered mobile devices run up to 50 concurrent SW use cases— from always-on Co-Pilots, high resolution multi-camera use cases, ultra-low-latency AR powered gaming, real-time multi-language translation.
- This leads to unprecedented SoC complexity from one generation to another leading to complex silicon debug
- Yet modern silicon debug is bottlenecked by:
  - Exploding Data Volume: Real-world scenarios (e.g., 120fps HDR video, 3D point clouds) overwhelm simulation.
  - Software Dependencies: Drivers, algorithms, and SW stacks
  - System-Level Gaps: Isolated IP debugs vs. real world multi-workload bugs

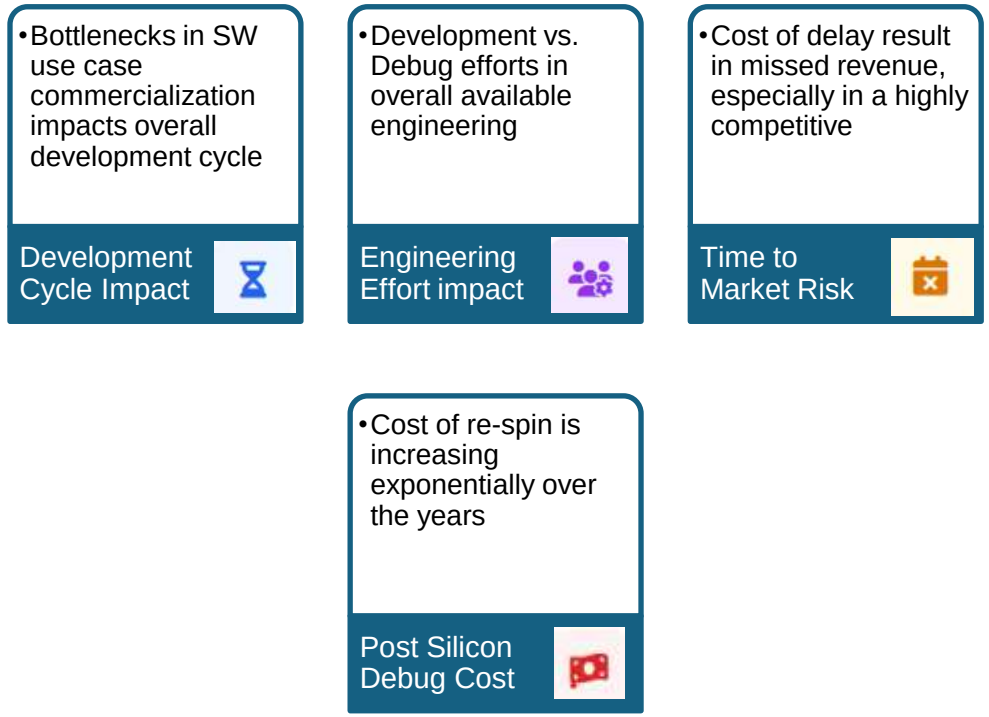
“80% of silicon escapes in mobile SoCs are system-level performance bugs—not functional errors.”



Problem Statement : Why Silicon Bugs are hard to reproduce in Pre-Silicon ?

Key Challenges on Silicon Debugs

- **Complex SoC interactions**
  - Silicon bugs often emerge only under long-running, highly concurrent SW use-case scenarios involving multi-subsystems
- **Post Silicon Visibility Gap**
  - Internal visibility drops drastically, hence root cause analysis becomes a black box with limited trace depth
- **Static nature of pre-silicon systems**
  - Existing systems fail to react to on-the-fly feedback such as backpressure, latency, leading to low bug reproduction rates



Problem Statement : Business Impact & Market Context

- **Business Impact**
  - Late discovery of performance & real-time failures
  - Excessive emulation cycles + senior engineer bandwidth
  - Schedule slips, delayed tape-out decisions, re-spin risk
- **Market Context**
  - Increased cost of pre and post silicon validation across the industry
  - Increased demand of FPGA based Emulation Platforms for system level verification and post silicon debug
  - Surging adoption rates for HW-Assisted Verification YoY

Pre-Silicon Testing

- $5.0B Market size in 2025
- Projected robust growth through driven by SoC complexity

Emulation Market

- $19.34B Projected value by 2030
- Growing from $11.73B in 2025. Critical for prototyping

HW Assisted Verification

- Surging adoption rate growth
- Shift towards software-first validation & full-SoC verification

Sources: DataInsights Market Reports (2025), MarketsandMarkets FPGA Forecast, Future Market Insights (FMI).

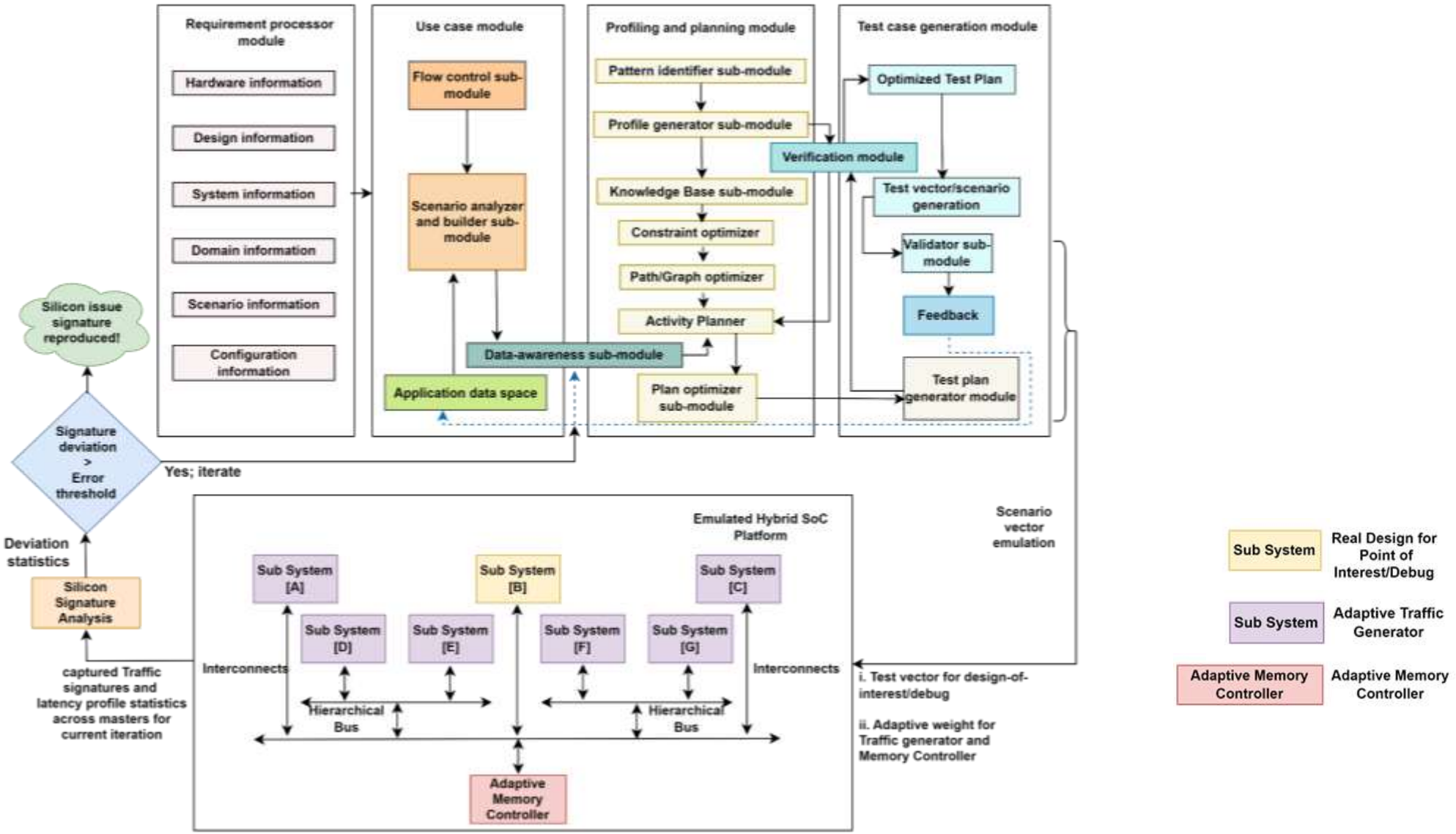
Problem Statement : Technical Challenges with Traditional Methods

- **Key technical challenges with Traditional Methods**
  - Scalability issues with limited concurrency, difficulty in modeling complex, simultaneous workloads realistically
  - Traffic generators are non-adaptive. Cannot respond to real, dynamic system contention
  - Memory & bus latency are uncontrolled. No per-master RD/WR latency fidelity, no peak modeling. No closed loop adaptative based on real silicon behavior
  - Stimulus tuning is manual, non adaptive and slow. Weeks of trial-and-error to reproduce one silicon issue
  - Low fidelity pre-silicon replay. Many real silicon failures are not reproducible

- **Core Gap**
  - Non-deterministic silicon behavior is being approximated using deterministic, static pre-silicon platforms
  - Modern SoCs fail at the system level — but we debug them with static tools

Introduction to Our Solution

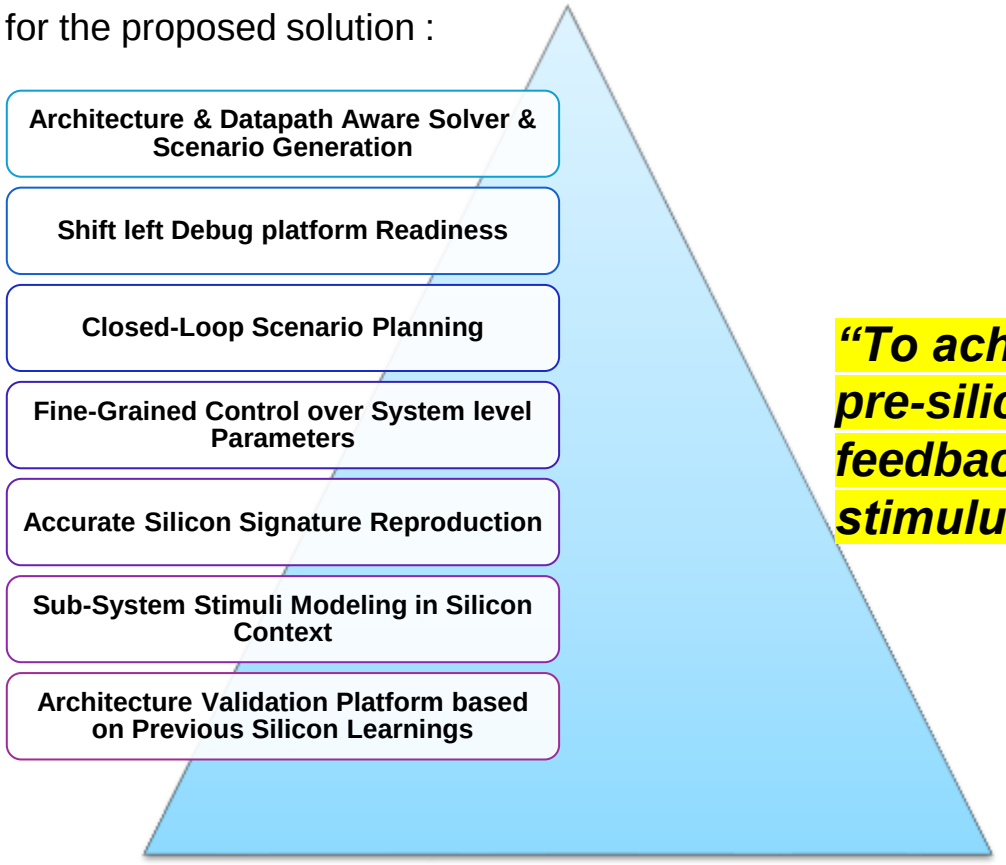
- To address the gap between real silicon behavior and pre-silicon debug platforms, we introduce a “**high-fidelity adaptive silicon replay framework”** that fundamentally changes how system-level failures are reproduced after tape-out.
- Instead of relying on static stimulus and fixed-latency models, our approach observes system behavior and adapts in real time, converging toward the exact traffic, timing, and latency signatures seen on silicon.
- Our solution enables pre-silicon environments to behave like silicon under stress—rather than approximating it.
- Key Components
  - At the core of this framework are fully synthesizable learning based adaptive traffic generators as well as adaptive bus modeling engine, designed to run natively on emulation platforms at speed.
  - By moving intelligence into synthesizable hardware, the solution eliminates costly host-emulator handshakes and drastically reduces the time to reach debug-ready waveforms.
  - The adaptive hardware layer is orchestrated by a closed-loop, agentic AI system that automates constraint solving, stimulus refinement, and convergence detection.
  - We present a fully-synthesizable Model Context Protocol (MCP) compliant hybrid SoC verification/validation system using multimodal Agentic AI, leveraging adaptive traffic generation and multi-master memory control for close real-time simulation.
- From a business perspective, this translates to “***faster issue convergence, fewer reproduction iterations, and earlier risk detection, directly improving time-to-market and reducing silicon re-spin risk”.***



Block diagram of our proposed “adaptive silicon replay platform”

Introduction to Our Solution – Core Objectives

Following are the core objectives for the proposed solution :



"To achieve high-fidelity silicon replay in pre-silicon emulation by closing the feedback loop between observation and stimulus"

Introduction to Our Solution – Why our Solution is different ?

Intelligent Adaptation using Agentic AI	Agentic AI powered stimuli refinement & tuning based on system response & continuous optimization using closed-loop feedback
Dynamic Knobs	Adaptive Traffic Generator adjusts transaction rates and traffic pattern to converge on silicon bandwidth signature
Optimization Targets	Matches QoS and response latency distributions to trigger rare bugs

Adaptive Traffic Generator

Existing Approaches : **Static Traffic & Behavioral Mismatch**

❌ **Traditional Traffic Generators**

- Fixed patterns and priorities
- No awareness of system response
- Manual tuning to hit bandwidth targets
- Lacks Self configurability

⚠ **Reality on Silicon**

- Dynamic contention
- Variable acceptance and latency
- Non-deterministic behavior

🎯 **Core Problem**

- Static stimulus cannot recreate dynamic silicon behavior

Our Solution : "**Closed-Loop Learning on Emulation"**

☑ **Fully synthesizable, emulation-speed IP**

- Observes system response in real time
- Learns arbitration weights dynamically

🔄 **Closed-Loop Adaptation**

- Target → Observe → Learn → Adapt → Converge

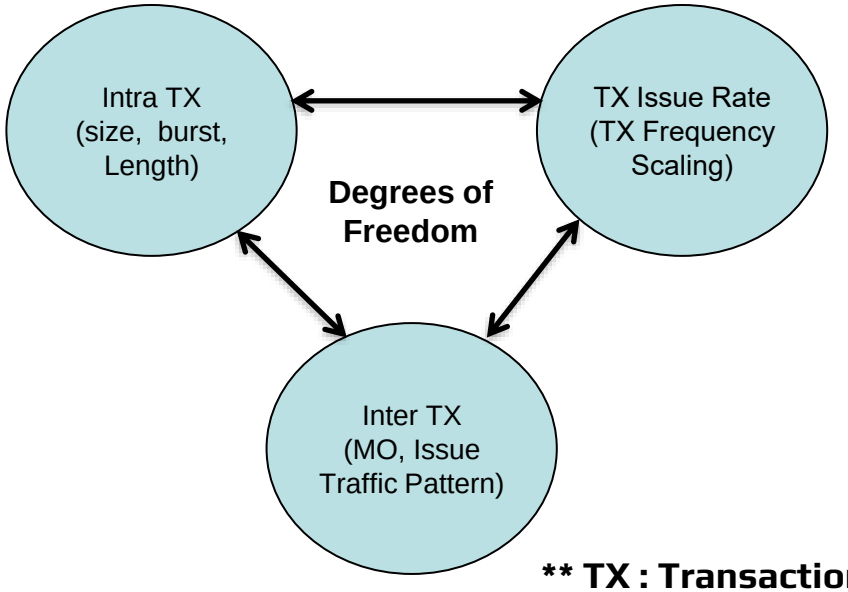
🚀 **Impact**

- Faster target convergence
- Fewer reproduction iterations
- Reduced debug effort

Adaptive Traffic Generator

🔧 Key Technical Capabilities

- Fully synthesizable traffic generation IP with 3 “degrees of freedom”
- Agentic AI powered Dynamic weighted arbitration at runtime
- Bandwidth target tracking across fine-grained time windows
- System-aware adaptation using acceptance rate & response latency
- Supports up to 32 logical clients/masters with protocol-compliant behavior



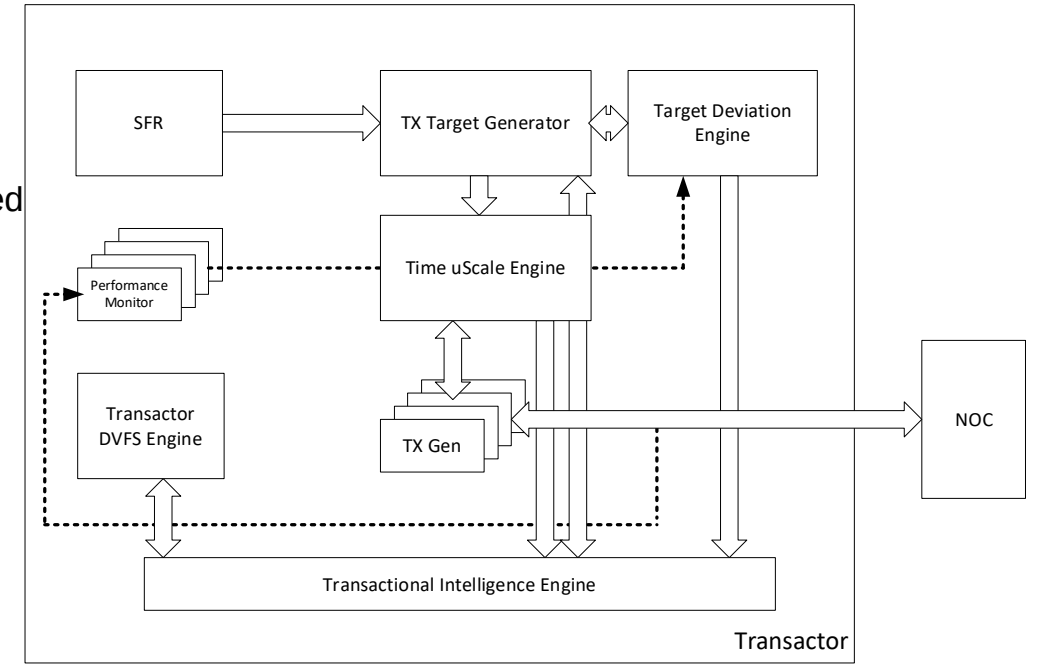
**** TX : Transaction**

📊 Measured Impact

- Faster convergence to silicon bandwidth signatures
- >80% reduction in manual stimulus tuning iterations
- Enables faithful replay of non-deterministic traffic scenarios
- Significantly improves pre-silicon confidence for system-level debug

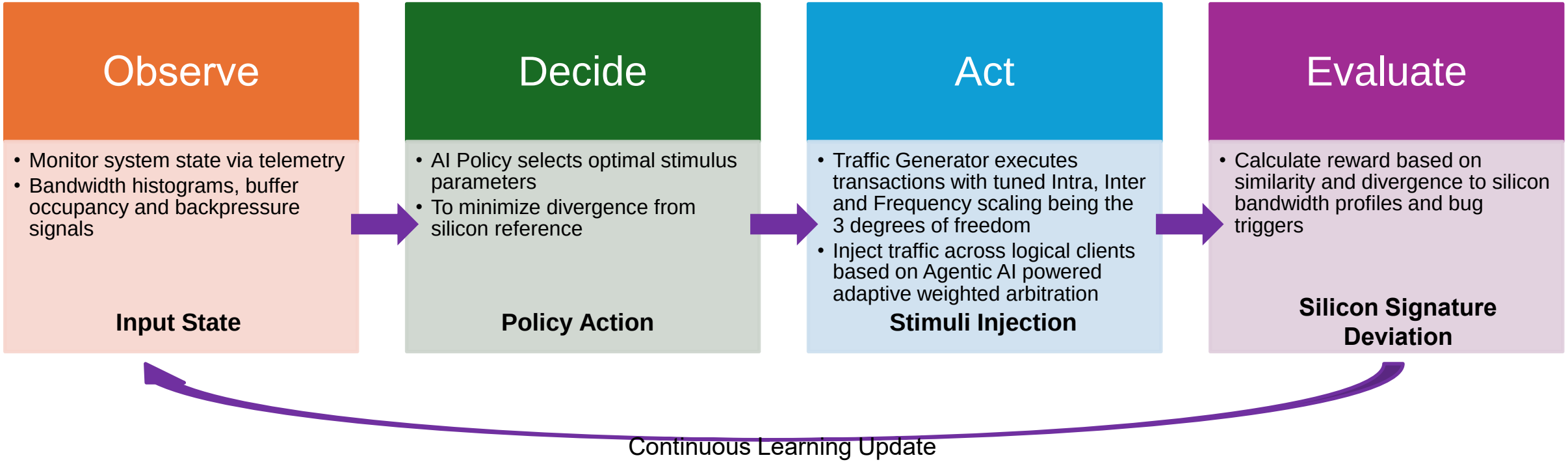
Adaptive Traffic Generator

- **TX Gen Cores** : Enables generation of transactions that comply with protocol specifications & handles feedback from rest of the system. Handles the 3 degrees of freedom
- **Micro-level Target Bandwidth Estimation Engine**: Enables the Traffic Generator to be self-configurable, and periodical config packet fetch from DRAM based on the closed loop feedback from the Agentic AI system.
- **Dynamic Adaptive Weight Generation Engine**: Enables generation of quantized weights across clients for priority based QoS arbitration within the Adaptive Traffic Generator using target deviation feedback
- **Client-wise Performance Monitor**: This block provides performance stats across clients that can help rest of the blocks in their functionality to collectively achieve the targets.
- **Target Deviation Engine**: Monitors logical client-wise BW target deviations at micro scale enables fetch of quantized weights via the Dynamic Adaptive Weight Generation Engine
- **DFS (Dynamic Frequency Scaling) Analysis Engine**: Tunes the degrees of freedom using Freq Scaling, as a last ditch effort if all other degrees of freedom fail.



Block Diagram of Adaptive Traffic Generator

Adaptive Traffic Generator : Closed Loop Mechanism



Multi-Objective
Simultaneously optimized for latency matching, throughput targets

Domain Priors
Protocol aware constraints for legal and valid traffic generation

Cross Sub System Sync
Agentic-AI coordinates b/w multiple TGs to create system level contention matching silicon behaviour

Emulation Ready
Fully synthesizable and light weight, will self configurability

Adaptive Bus Modelling Engine

Existing Approaches : **“Low Latency Fidelity Breaks Silicon Replay”**

❌ Conventional Memory Modeling Gaps

- Fixed or uncontrolled latency behavior
- No per-master RD/WR latency control
- No peak latency modeling
- Lacks Scalability

⚠ Reality on Silicon

- Latency varies per master and priority
- RT vs NRT traffic interference
- Rare latency spikes cause failures

🎯 Core Problem

Uncontrolled latency prevents faithful silicon replay

Our Solution : **“Closed-Loop Latency Convergence”**

☑ Fully Synthesizable Adaptive Bus Modelling Engine

- Emulation-speed operation
- No firmware or CPU dependency
- Per read/write average and peak latency modelling
- Scalable Architecture

🔄 Closed-Loop Adaptation

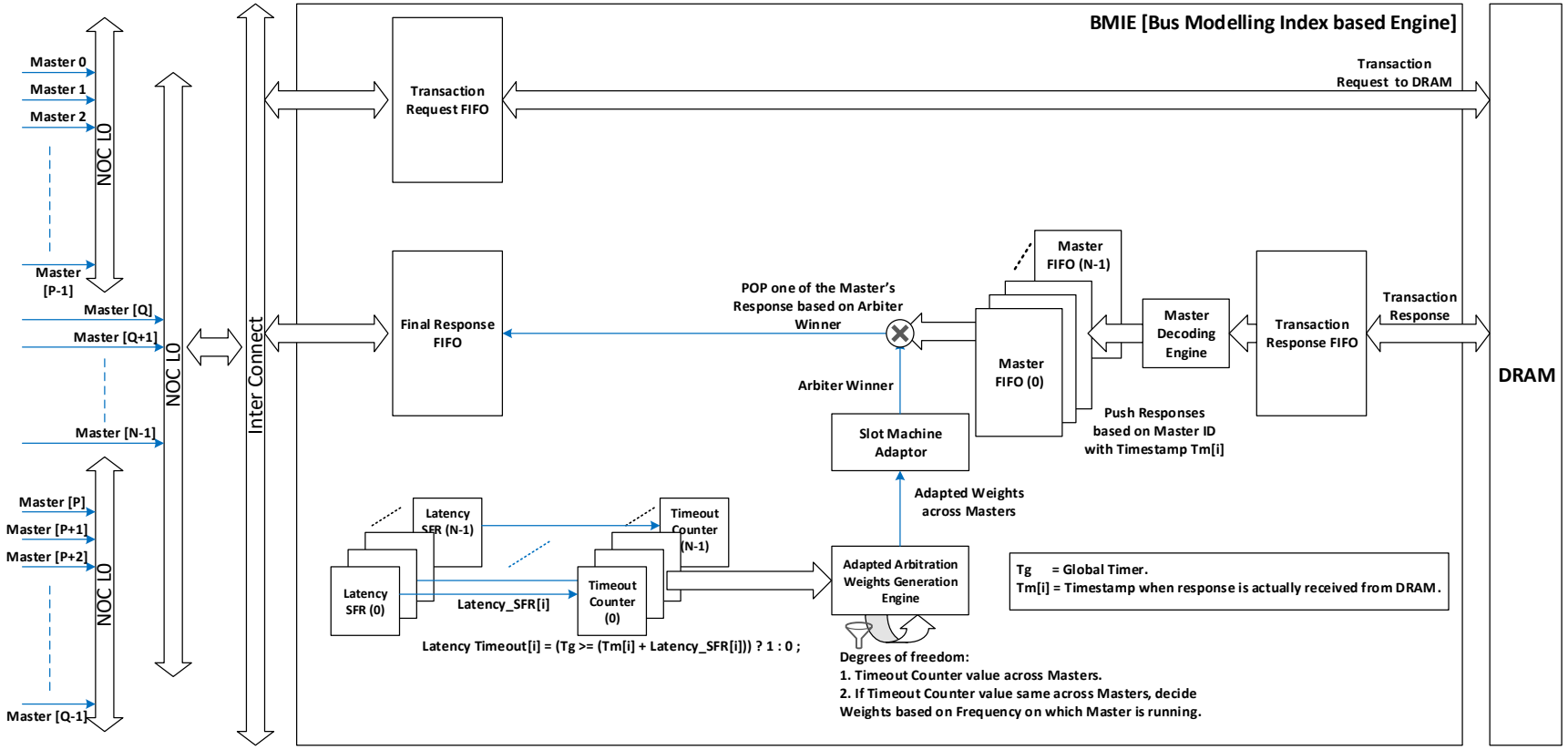
Observed latency → Error tracking → Arbiter tuning → Convergence

🚀 Impact

- Reproduces rare RT failures pre-silicon
- >15% improvement in latency accuracy
- Faster, higher-confidence debug

Adaptive Bus Modelling Engine

- **Master Decoding Engine**: Supports decoding up to 128 Masters which generate traffic based on the merged traffic at the final NOC hierarchy, with configurable decoding logic.
- **System Control Registers**: For Configurable independent SFRs per Master (up to 128 Masters) for WR and RD accesses, covering both average and peak latency values.
- **Slot Machine Arbiter Engine**: Weighted Slot machine arbiter across Masters based on adapted weights input that are calculated based on the dynamic system behavior.
- **Adaptive Arbitration Weight Generation Engine**: Uses adaptive weights from the Agentic AI closed loop system, for each Master and allocates arbitration scheme based on QoS.



Block Diagram of Adaptive Bus Modelling Engine

Adaptive Bus Modelling Engine

🔧 Key Technical Capabilities

- Independent latency control per master (128+ masters)
- Separate RD / WR average latency modeling
- Peak latency injection for corner cases
- Agentic AI-powered Adaptive QoS arbitration & scheduling logic

📊 Proven Results

- <1.25% latency error vs silicon
- Fewer replay iterations
- Enables system-level validation on pre-silicon

Agentic AI based Constraints Solver & Vector Generator

Existing Methods : **“Manual Stimuli Refinement”**

❌ Traditional Debug & Stimulus Generation

- Manual constraint tuning and stimulus iteration
- Heavy dependence on expert intuition
- Poor scalability across SoC size and complexity

⚠ Reality of Modern SoCs

- Thousands of interacting parameters
- Non-deterministic system behavior
- Rare failures with no clear trigger

🎯 Core Gap

Human-driven debug loops cannot scale to system-level silicon replay

Our Solution : **“Agentic AI Orchestrated System”**

☑ Agentic AI Control Plane (MCP-Compliant)

Orchestrator Agent

- Coordinates tasks and convergence flow

Specialized Agents

- Constraint Agent – constraint simplification & solving
- Graph Agent – dependency & path analysis
- Generator Agent – test vector generation
- Test Agent – validation & convergence checks

All agents share context via MCP servers for safe, scalable orchestration

Agentic AI based Constraints Solver & Vector Generator

🔄 Closed-Loop Execution

Constraints → Generate Stimulus → Execute on Emulation Platform → Observe Signatures → Refine Constraints → Converge

🎯 What the AI Learns

- Effective constraint combinations across complex workloads
- Traffic and latency sensitivity patterns
- Signature convergence thresholds

💼 Impact

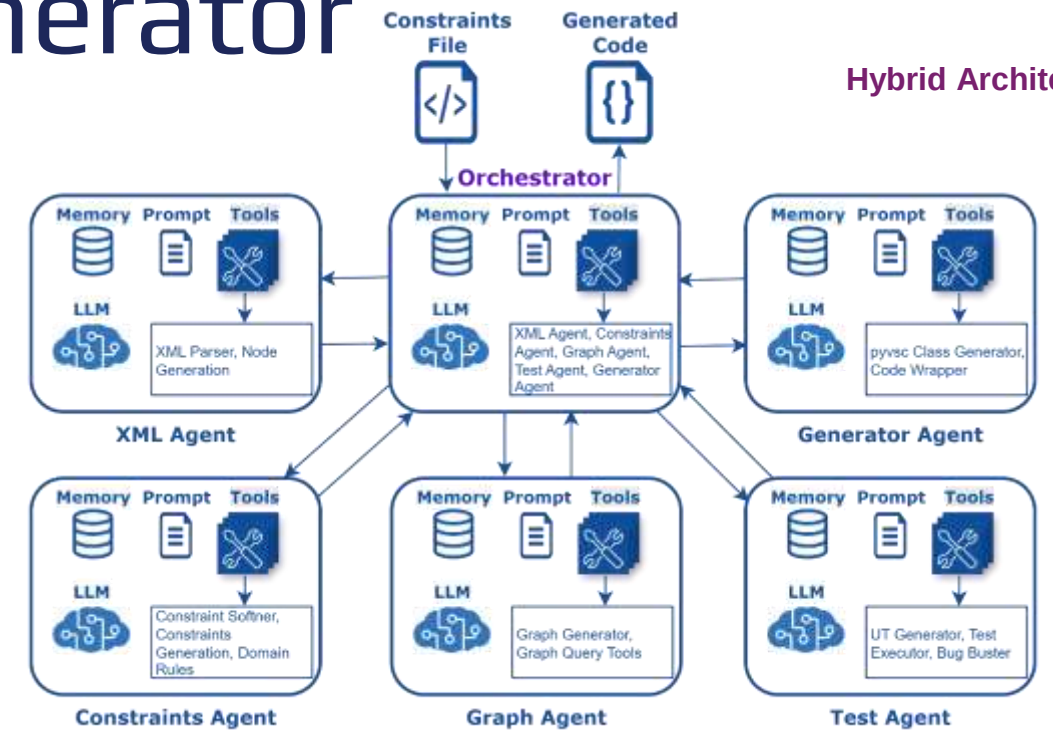
- >80% reduction in reproduction iterations
- Faster debug turnaround
- Knowledge captured and reused across projects

Agentic AI based Constraints Solver & Vector Generator

Hybrid Architecture with GenAI and SMT Solvers

- Parses XML input file.
- Extract configuration details
- Construct structured node tree

- Simplifies complex constraint definitions using domain rules.
- Converts into a required SMT format.



- Compiles constraint functions into an executable scripts
- Generates the test vectors with randomized seeds

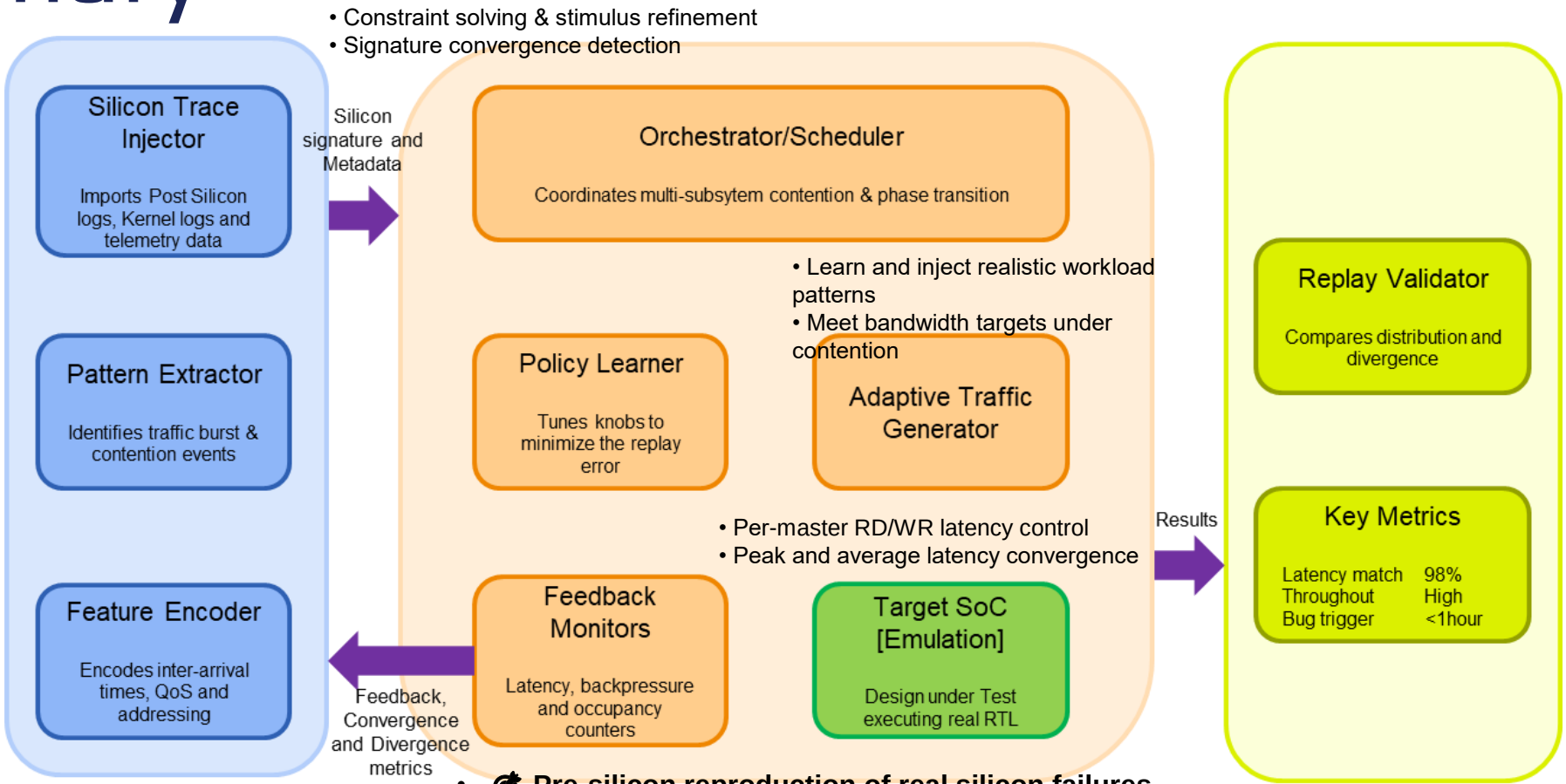
- Generates unit test for each constraint.
- Validates satisfiability using SMT library rules.
- Automates error detection.

This modular, agentic framework combines domain specific processing with AI-powered constraint mapping to automate and enhance test vector generation.

- Constructs subgraphs.
- Checks for cyclic dependencies.
- Identifies and resolves cyclic dependencies.

Figure : The information flow across various agents for the core constraint solver in the Agentic AI framework

Adaptive Silicon Replay Platform: Solution Summary



- 🎯 **Pre-silicon reproduction of real silicon failures**
- 💼 **Faster debug, fewer iterations, reduced re-spin risk**

Real Silicon Failures Reproduced : From Silicon Escape to Pre-Silicon Catch

Case 1: Camera Buffer Overflow

- **Silicon Symptom**: Frame drops during camera preview while gaming
- **Root Cause:** GPU bandwidth starvation → Camera ISP FIFO overflow
- **Traditional Debug**: 6+ weeks, no reproduction
- **Our Platform**: < 1 day, full system replay

Case 2: Multi-Frame Processing Hang

- **Silicon Symptom**: Camera hang once every 6 days
- **Root Cause**: Cache prefetch bug + high DRAM latency for reads but not writes
- **Traditional Debug**: Intermittent, impossible to trigger
- **Our Platform**: < 1 day, deterministic reproduction

Case 3: AI Inference Stall During Recording

- **Silicon Symptom**: NPU hang during extended video capture
- **Root Cause**: DVFS lock conflict + thermal throttling
- **Traditional Debug**: Post-silicon only, lab debug required
- **Our Platform**: Pre-silicon, auto-triggered with <2% timing error

Key Insights:

· ☑ No prior root cause knowledge needed
· ☑ Bare-metal replay without CPU/FW dependency
· ☑ Deterministic reproduction of intermittent bugs
· ☑ System-level interactions accurately modeled

Turning "impossible to reproduce" into "replayed in hours"

Performance Benchmarking vs. S.O.T.A : Quantifiable Speed & Accuracy Gains

Key Performance Indicators	Traditional approach	Our approach	Comparative metrics
Iterations for silicon signature convergence [ISP + Display Processor Chain for Preview @ FHD 60]	~34	~4	**~9x faster convergence**
Latency profile modelling accuracy [Display Processor Preview FHD @ 60fps]	85%	98%	**>15% gain in accuracy**
Time taken for debug-ready waveform availability [ISP + Display Processor Chain for Preview @ FHD 60]	16-18 hours	20-22 minutes	**~50x faster**

Key Performance Indicators (KPI) comparison

Solving type	EDA Solvers	Our approach
Single Camera IP Solving	192 sec	**6 sec**
Partial Camera Chain Solving [ISP Pre Processing 16MP]	643 sec	**12 sec**
Full Camera Chain [ISP + Display Processor Chain for Preview]	981 sec	**13 sec**

Comparative performance evaluation of various solvers

Technical Drivers

· AI-driven adaptive learning & Faster convergence to failure signatures
· Closed-loop optimization & Data path aware solving
· Minimal host-emulator data exchange

Results : Adaptive Learning in Action (Adaptive Traffic Generator)

AI-Driven Adaptation by Traffic Generators : Learning Behavior

1. Initial Phase: Client 3 gets highest priority (high packets + high latency), then others catch up

2. Adaptive Phase: Weights adjust based on:

   · Target achievement rate

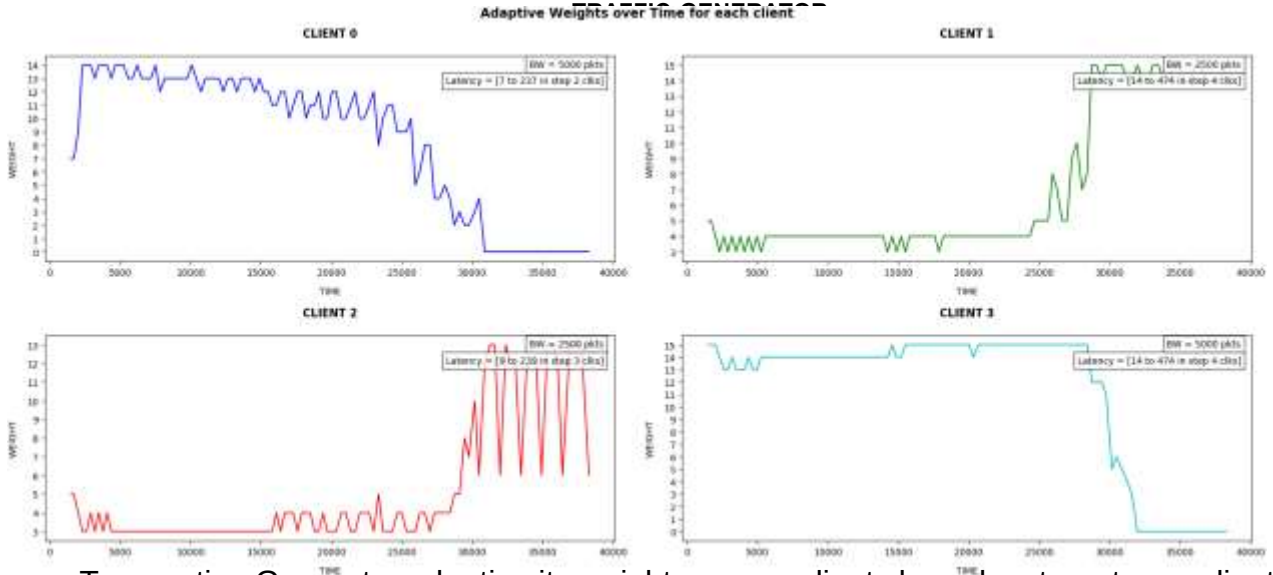
   · System response patterns

   · Competing client behavior

3. Convergence: All clients meet targets within 5% target error margin

Client Number	Number of transactions targeted (in packets)	Number of Clock cycles available to reach target (in clock cycles)	Latency seen for the responses for the sent transactions (in clock cycles)
0	5000	40000	random range(7, 237)
1	2500	40000	random range(14, 474)
2	2500	40000	random range(9, 239)
3	5000	40000	random range(14, 474)

SYSTEM TEST SCENARIO BEING EMULATED TO ANALYZE THE ADAPTATION OF TRAFFIC GENERATOR



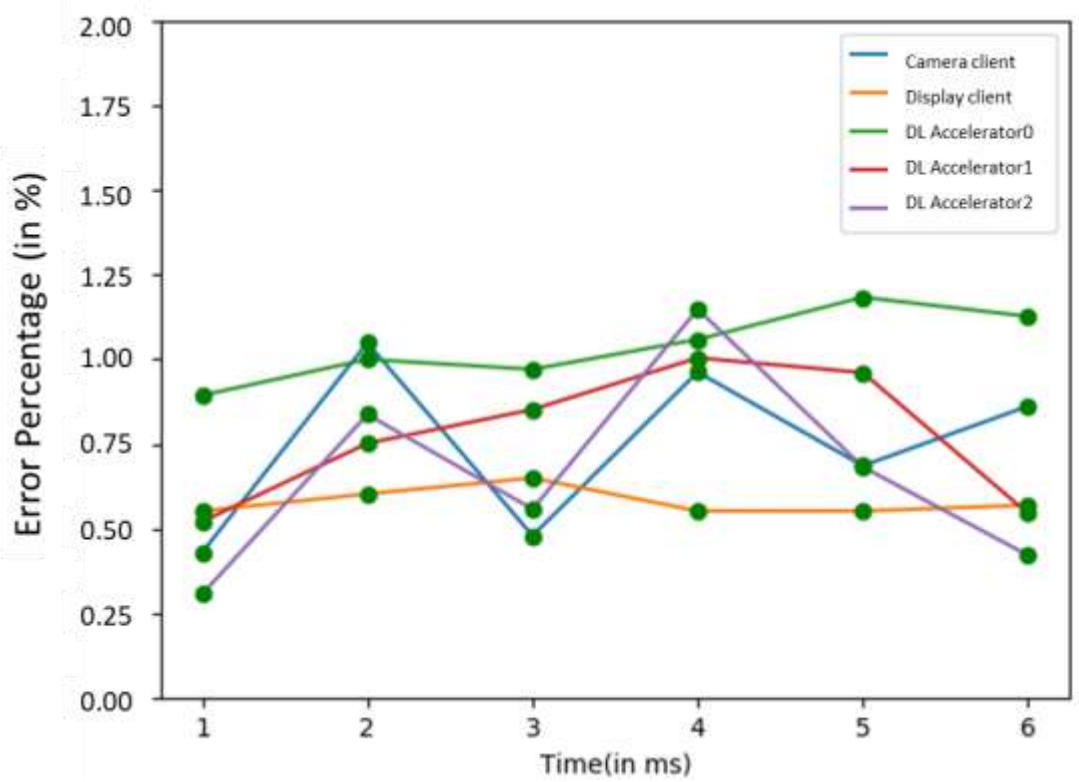
Transaction Generator adapting its weights across clients based on targets per client as well as ad-hoc system response behavior

Results : Adaptive Learning in Action (Adaptive Traffic Generator)

Client Number	Number of transactions targeted (in packets)	Number of Clock cycles available to reach target (in clock cycles)	Latency seen for the responses for the sent transactions (in clock cycles)
0	5000	40000	random range(7, 237)
1	2500	40000	random range(14, 474)
2	2500	40000	random range(9, 239)
3	5000	40000	random range(14, 474)

Transaction Generator adapting its weights across clients based on targets per client as well as adhoc system response behavior

Results : Adaptive Learning in Action (Adaptive Bus Modelling Engine)



Adaptive Bus Modeling Engine Accuracy : Achieved Results:

- <1.25% error per master (read and write independently)
- Peak latency injection capability for DRAM refresh modeling
- Per-master QoS control for real-time vs. non-real-time traffic

Results of how different Masters in Adaptive Bus Modelling Engine are scheduled by the arbiter using adaptive weights generated from history (including error statistics) and learns itself to adapt in terms of future weights calculation to meet the latency targets converging with minimal error margin (<1.25%) w.r.t. silicon dumps

Error % seen across Masters for one of the Real Silicon reproduced issue almost <1.25%

Conclusion and Future Work

Research Contributions & Conclusions

- Closed-Loop Adaptive System
- Policy Learning Mechanism
- Multi-Sub System Coordination to recreate complex system-level contention.
- Demonstrated validated Silicon Replay platform

Business Impact

- Faster Feature Rollout
- Reduced Debug Cost
- Empowering IP Team
- Revenue Acceleration

Future Roadmap

- Active learning for Rare Events
- Hybrid Trace Generation
- HW/SW Co-Verification
- Expand methodology to broader Domains

Thank You