2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
OCTOBER 14-15, 2025

A Novel Configurable UVM Architecture To Unlock 1.6T Ethernet Verification

Sameh El-Ashry

Cadence Design Systems

cadence®

accellera SYSTEMS INITIATIVE

Agenda

01

Overview & Motivation

Industry trends and verification challenges

02

Related Work

Prior research and gaps in automation

03

Ethernet Controller

RTL architecture and configurations

04

Dynamic UVM Generation

Templating and automation flow

05

Case Studies & Results

Metrics, best practices, and lessons learned

The Challenge: Ethernet Verification Complexity[1]

The problem

- Low-speed and High-speed Ethernet verification (10Mb → 1.6T) is complex.
- Multiple RTL configurations (RTL defines, RTL builder tool).
- Manual UVM adaptation → time-consuming, error-prone.
- Goal: fully automated, UVM testbench-matching DUT generation.

The Challenge: Ethernet Verification Complexity[2]

Key Motivations

- Increasing Bandwidth Demands.
- Verification Complexity.
- Reusable UVM Environments.
- Automation and Efficiency.

Related Work: Filling the Gap

Prior Research

Automatic UVM generation from assertions and functional verification models.

Critical Gap

Existing methods do not handle RTL defines or macros during UVM generation.

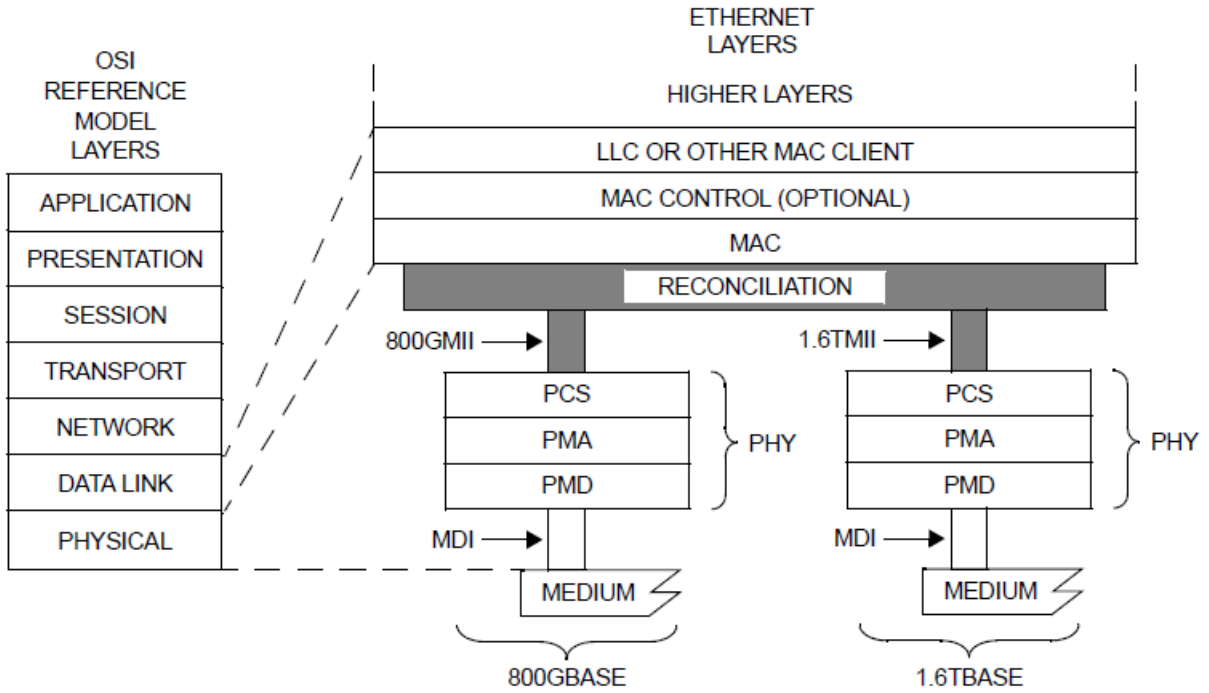
Our Innovation

Uniquely supports RTL defines → UVM generation for configurable designs.

Ethernet Controller RTL Overview

- The controller consists of multiple layers: AF → UEC → MAC → PCS → FEC → PMA
- RTL Builder tool generates customized RTL + defines.
- RTL Configurations include:

- **SerDes Widths**
  - Variable data path configurations
- **AF/MII Widths**
  - Configurable interface parameters
- **Optional UEC Layer**
  - Credit-based flow control
- **PCS-Only Modes**
  - Simplified configurations

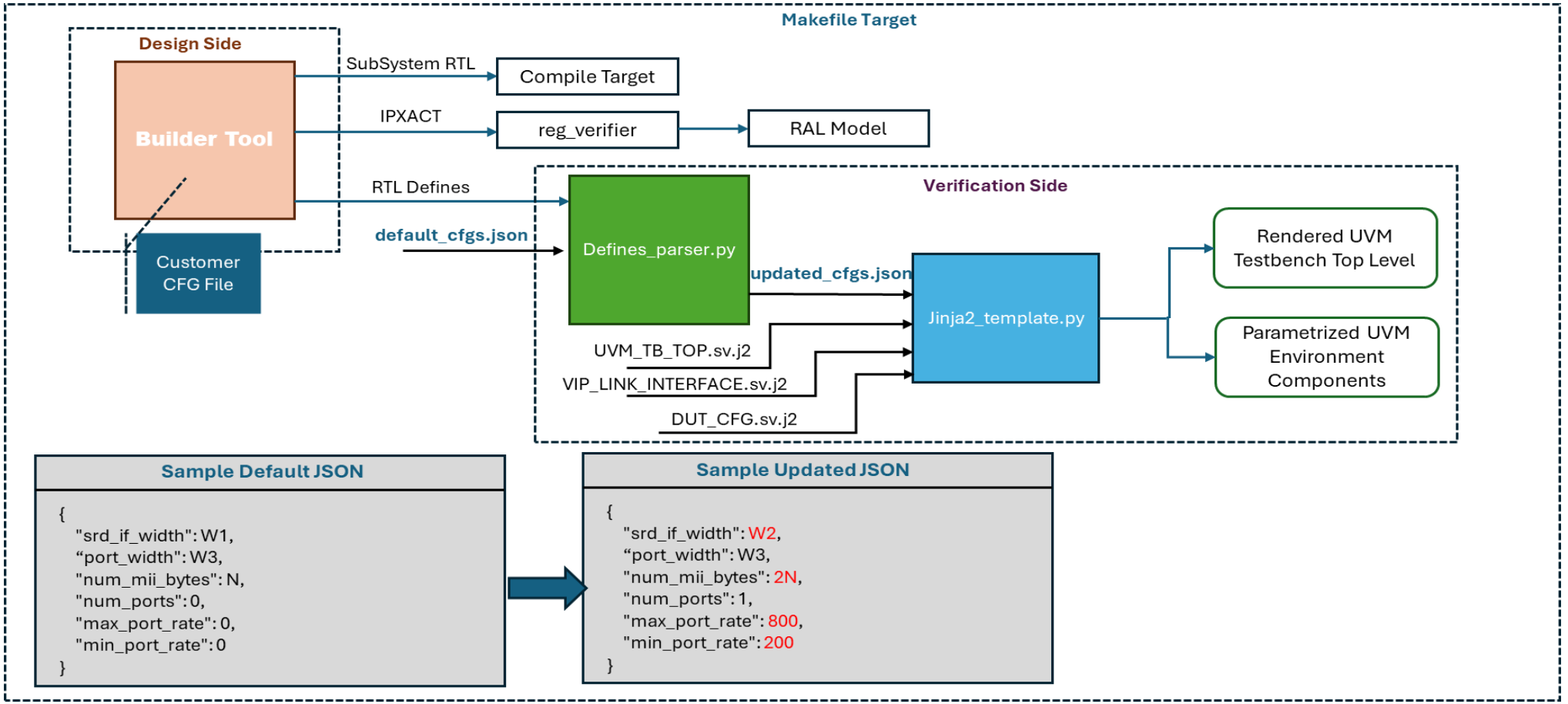


*IEEE Std 802.3

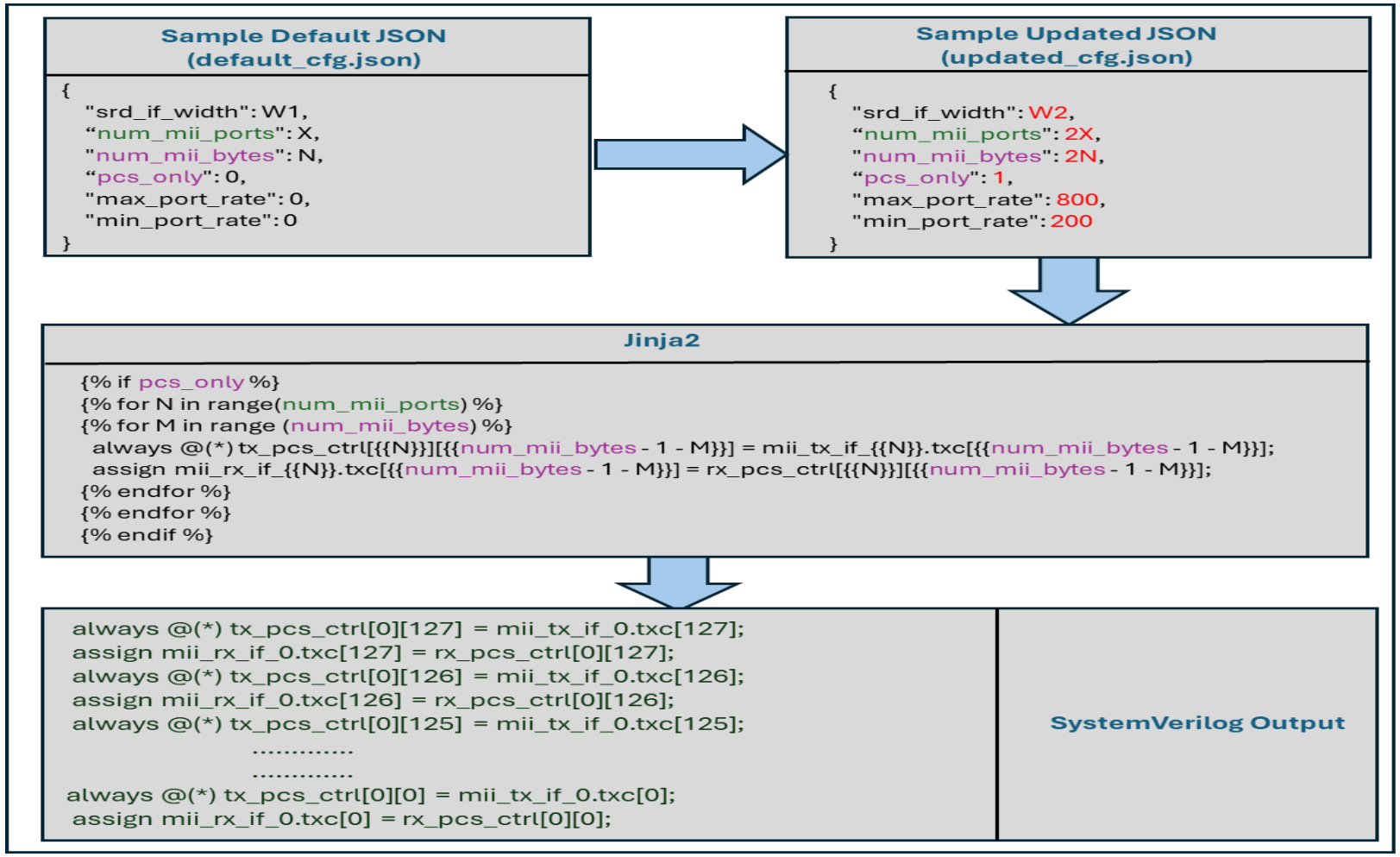
Challenges in Configurable Verification

- Large configuration space → exponential testbench variants
- Time to rewrite top-level files for each config
- Interface mismatches cause debug cycles
- Resource limits → need automation

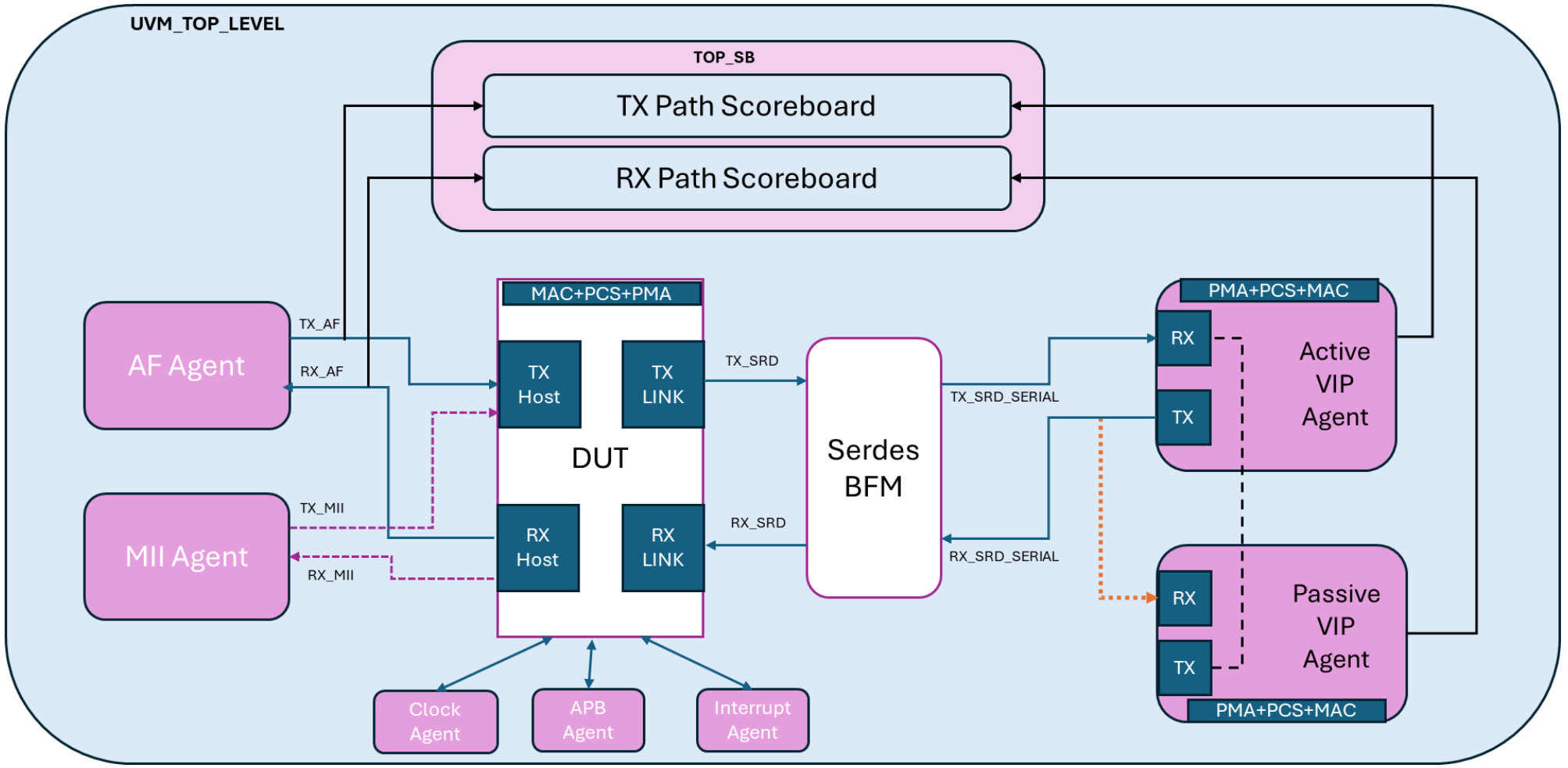
Dynamic UVM Architecture Generation (Automation Flow)



Templating UVM Components with Configuration Input

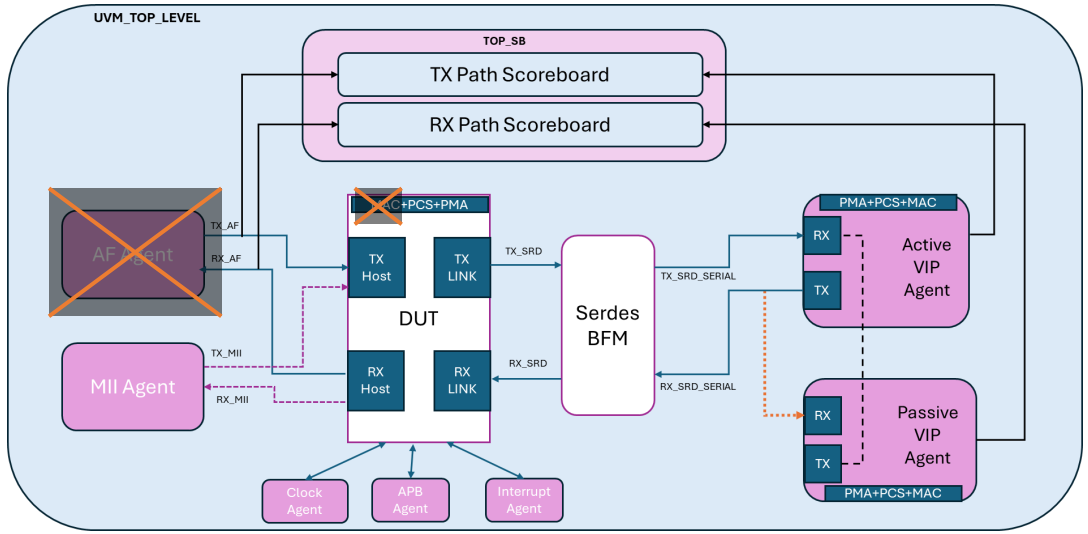


UVM Environment – Full Controller



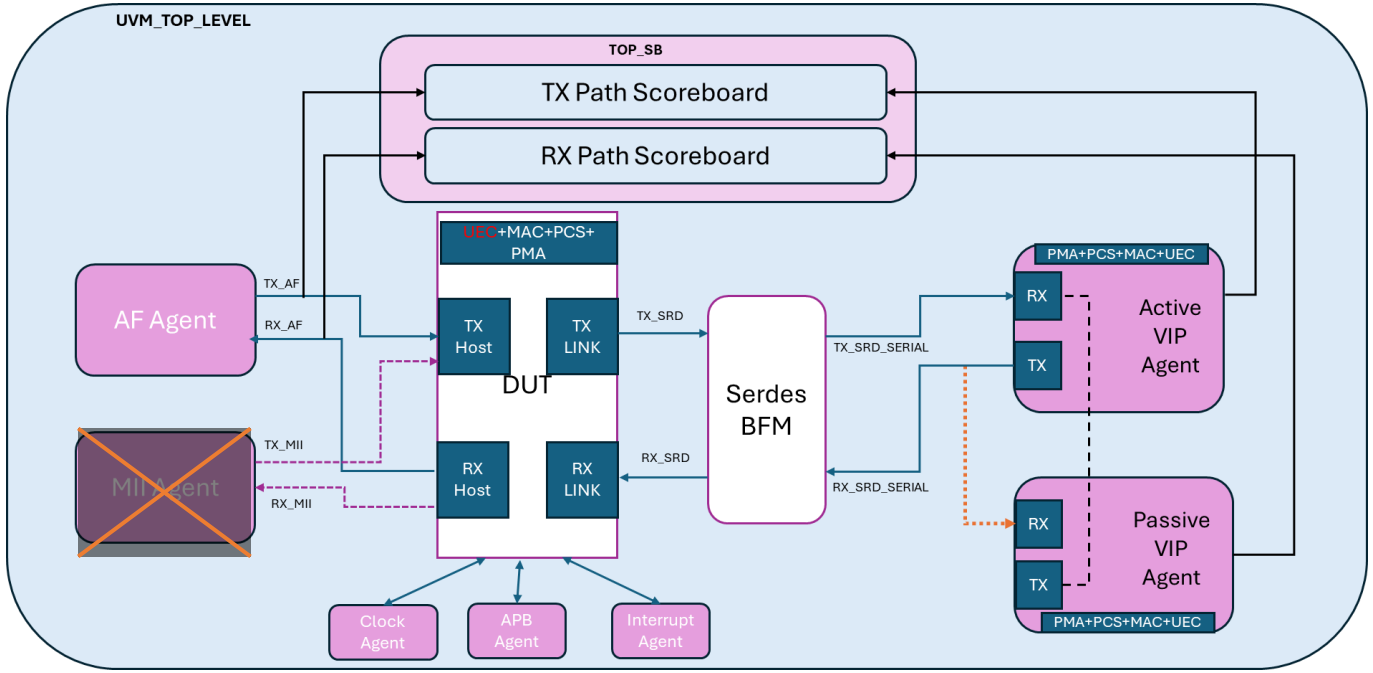
PCS-Only Mode

- AF agent replaced by MII agent
- Virtual sequencer drives PCS-only tests
- Majority of sequences reused from full controller



UEC Extensions

- Credit-based flow control
- LLR replay buffer
- Stimulus adapted from AF agent scenarios



Time Savings from Automation Flow

20	<1	3-4	~0%
Days Manual	Minute Automated	Days Setup	Error Rate
Time required for 10 configurations using manual approach	Time for same configurations with automation flow	One-time template setup effort	Negligible errors vs. frequent human mistakes

- The automated flow achieves 100% instantiation accuracy with zero rework when switching between configurations. New configs are essentially free after initial template investment.

Best Practices

- Separate templated vs parameterized components.
- Keep JSON as single source of truth.
- Automate RAL generation (reg-verifier).
- Use Continuous Integration (CI) pipeline for nightly configs regressions.
- Gradually expand automation scope.

Results and Challenges

METRIC	MANUAL FLOW	PROPOSED AUTOMATED FLOW
Top-level testbench creation time (per config)	1–2 days	seconds
Human errors in port/interface connections	High (frequent debug cycles)	Negligible
Effort to switch between full and PCS-only mode	Manual rework	Zero rework
Number of supported DUT configurations	1–2 feasible manually	10+ handled with ease
Reuse of sequences between configurations	Partial	High, via virtual sequencer
Required testbench maintenance	High (per variant)	Centralized and minimal
Parameterized UVM agent/scoreboard support	Manual setup per config	Pre-built and reusable

Scalability to Other IPs

- Approach can be reused for:
  - Custom memory controllers (with varying data widths, burst sizes, or ECC enablement).
  - PCIe Controller (e.g., Configurable as Root Port or Endpoint with varying lane width, speed and data path width).
  - SoC peripherals (UART, SPI, I2C with parameterized FIFO depths, number of instances, etc.).
  - DMA engines (supporting multiple channels, memory-mapped, and streaming modes).
  - AI/ML accelerators (with changing compute array sizes, interfaces, or hierarchy depth).

Lessons Learned

- Automation saves effort but requires upfront design.
- Templates must be carefully tested.
- Parameterization complements templating.
- Collaboration with designers is key.

Conclusion

- Automated flow enables UVM testbenches-matching DUT
- Time savings: days → seconds
- Reusable across configs and IP types
- Paves way for scalable DV of complex controllers

Questions