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

Breaking the wait: Real-Time Post-Processing for Data Integrity Verification in SerDes Systems

Suresh Gandhi S
Chandana K N

intel

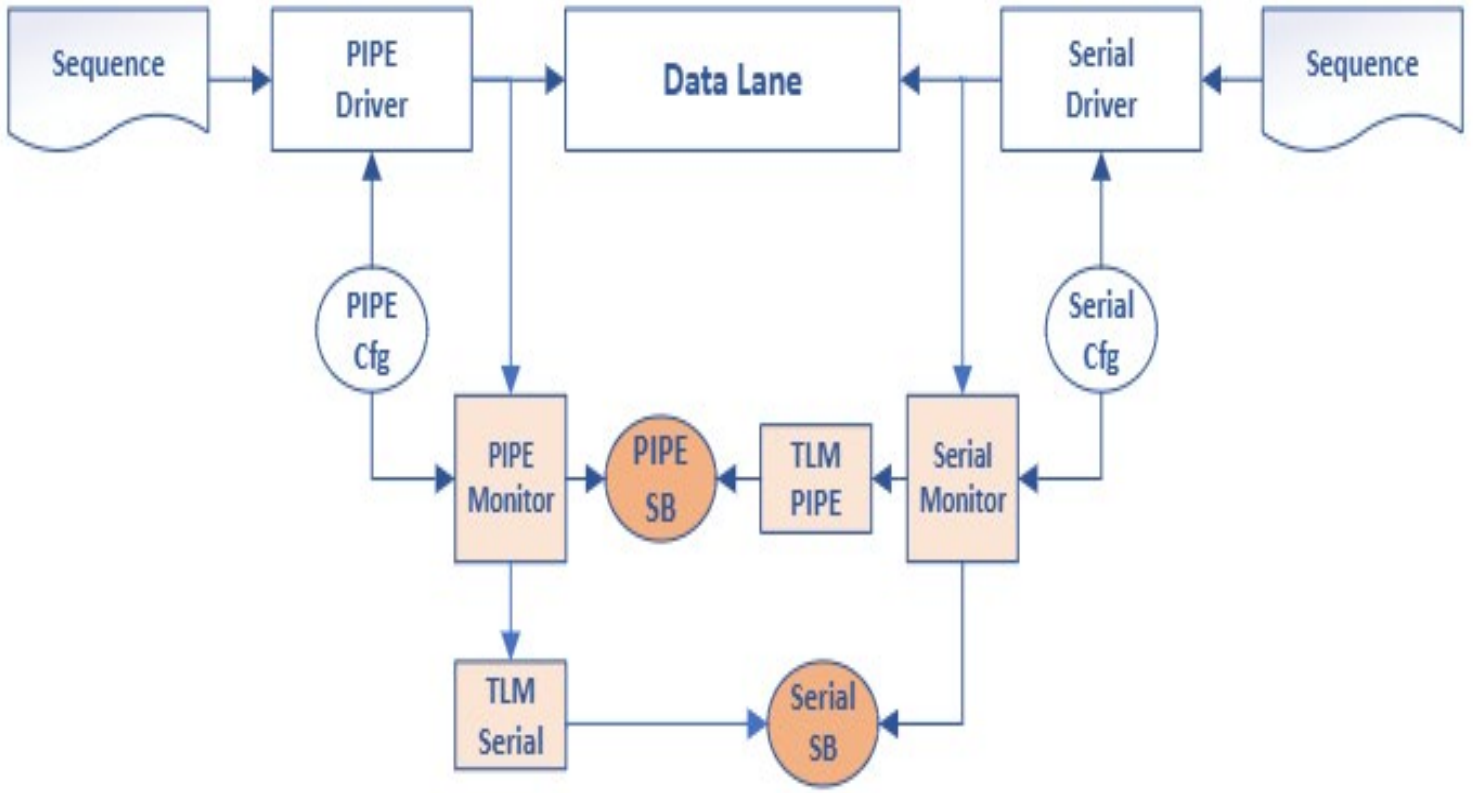
Challenges in SerDes Verification

Modular PHY	• Single Design that supports increasing data rates, protocol diversity, complex modulation schemes.
Reference Models	• Reference models for verification are highly protocol-dependent , requiring extensive development, maintenance to support multiple standards.
Debugging Bit-Level Mismatches	• Debugging continuous data streams without clear boundaries is tedious and time-consuming.
Need for Flexible Verification	• There is a need for a simpler SerDes data integrity verification compared to scoreboard based on reference models

Reference model-based checks

TLM Reference Model	Data Integrity Checks	UVM based TLM scoreboards – monitor samples required interface signals every clock cycle, making it more simulator expensive
	Control Flow Checks	Causal Checks – specific acknowledgements follow specific requests *M2P<->P2M, RXStandby -> RxStandbyStatus* Latency Checks - time intervals between events adhere to the defined timing constraints *L* entry to exit, preset coeff to coeff update* Event flow Checks - correct order of events within an operation *L* entry to exit to Phy status*

TLM Based scoreboard



Python-Based Data Integrity Verification

Independent Data Capture

Captures raw TX and RX data streams into text files independent of DUV and scoreboards, enabling flexible verification.

Configurable Parameters

Supports multiple encoding schemes like 8b10b, 128b130b, PAM and NRZ.

Tolerance handling

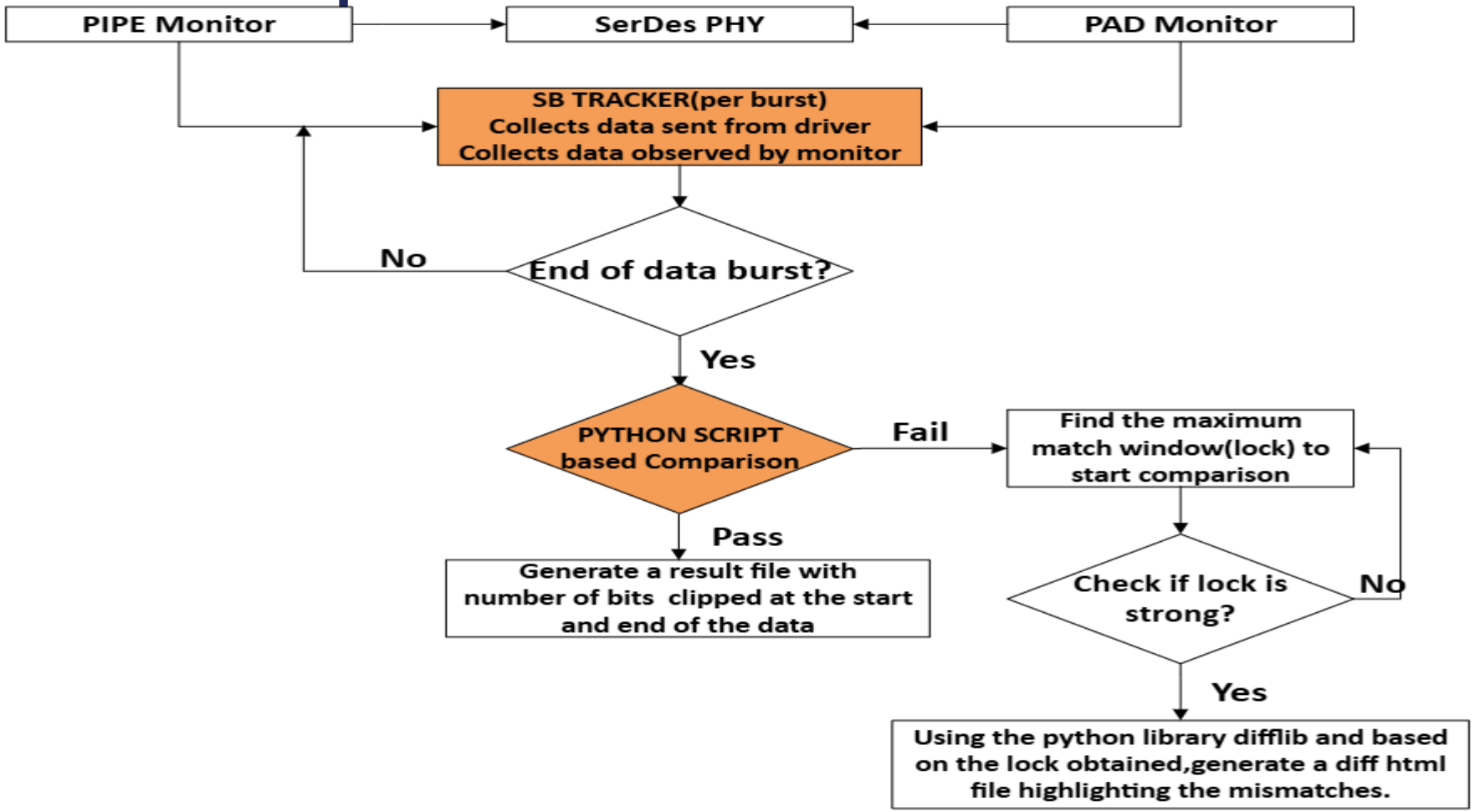
Real-Time Intermediate Results

Enables intermediate results during execution with customizable granularity at burst, symbol, or lane level.

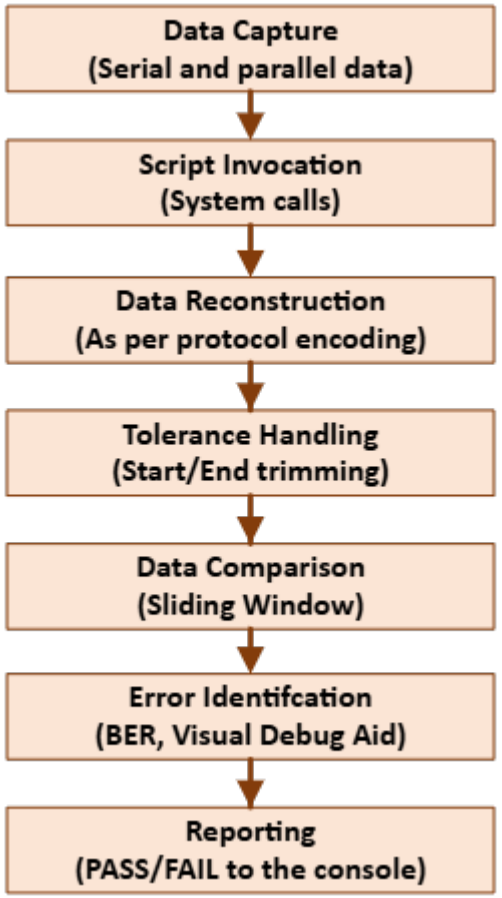
Visual Debugging with Python

Generates HTML diff files using Python libraries for efficient visual debugging and mismatch pinpointing.

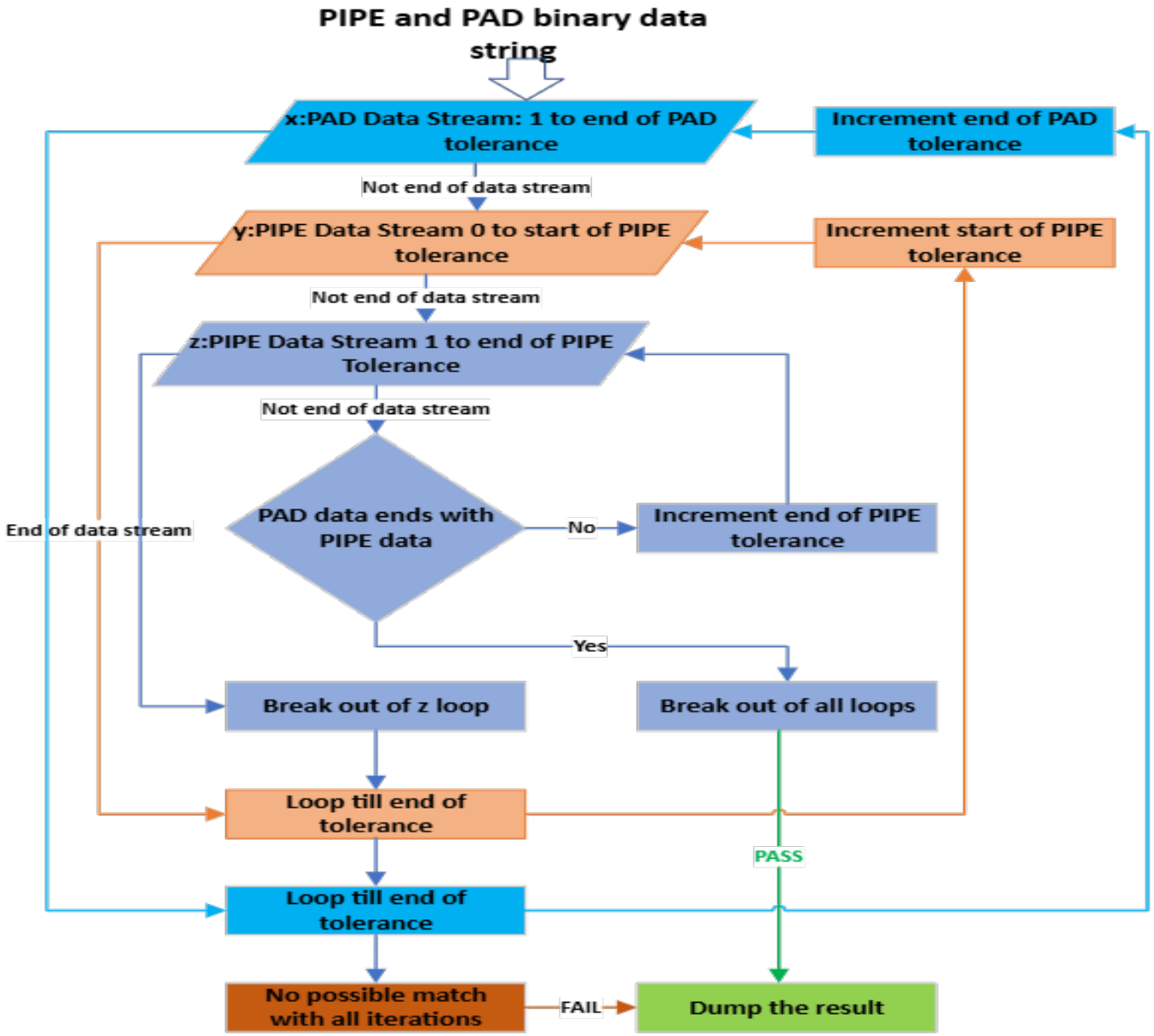
Python Script based Scoreboard



Post Processing Flow



Data Comparison Mechanism



Post Processing scoreboard

Easy bug fix : if the bug is in the scoreboard, there is no need to run the entire regression again

Python: complex code in fewer lines – easy to read, maintain

Wait time: Need to wait till the sim is over to get intermediate results

Sliding Window Mechanism

- Moves a fixed-size window across the data stream to find the best match.
- Calculates similarity ratio (using Python's SequenceMatcher).
- Supports coarse and fine granularity (per burst, symbol, lane).
- Enables robust, bit-accurate comparison even with protocol overhead.
- As an example, consider two streams to be compared as:

  TX data: ABCDEF, RX data: XXABCDEFYY

  A direct comparison would fail due to the extra XX at the start and YY at the end.

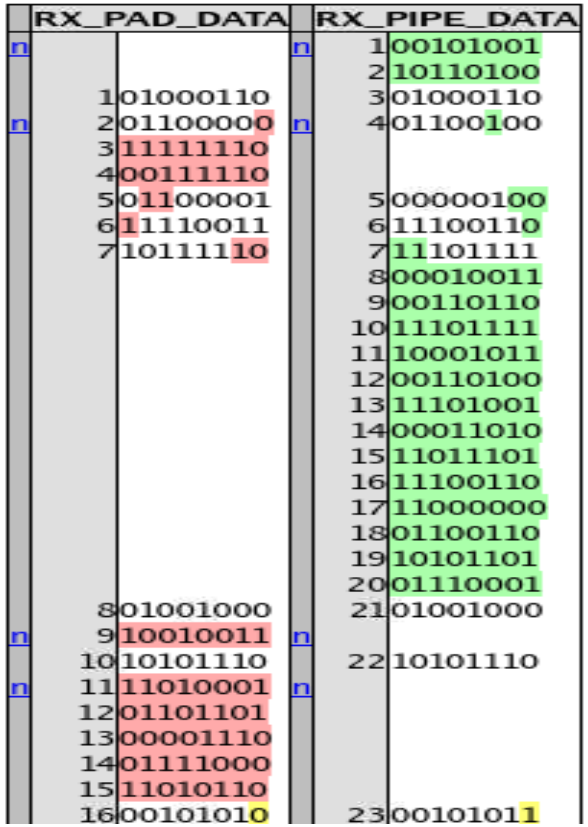
  With a sliding window of size 6:

  - *Window 1: XXABCD (no match)*
  - *Window 2: XABCDE (no match)*
  - *Window 3: ABCDEF (match!)*

  The script would report a pass at window position 3.

Visual Debugging Tool

- Side-by-side comparison of TX and RX data.
- Mismatches highlighted for rapid identification.
- Provides navigation links for next failure.
- Includes context: burst, symbol, bit position.
- Protocol-agnostic; works for any encoding.



	RX_PAD_DATA		RX_PIPE_DATA
n		n	1 00101001
			2 10110100
	1 01000110		3 01000110
n	2 01100000	n	4 01100100
	3 11111110		
	4 00111110		
	5 01100001		5 00000100
	6 11110011		6 11100110
	7 10111110		7 11101111
			8 00010011
			9 00110110
			10 11101111
			11 10001011
			12 00110100
			13 11101001
			14 00011010
			15 11011101
			16 11100110
			17 11000000
			18 01100110
			19 10101101
			20 01110001
	8 01001000		21 01001000
n	9 10010011	n	
	10 10101110		22 10101110
n	11 11010001	n	
	12 01101101		
	13 00001110		
	14 01111000		
	15 11010110		
	16 00101010		23 00101011

TLM vs Python based Scoreboard

Feature	TLM based Scoreboard	HTML Diff Scoreboard
Bit-level pinpointing	Manual, slow	Automatic, visual
Protocol dependency	High	Low (agnostic)
Debug time	Long	Short
Usability	Low	High
Realtime feedback	Yes, by default	User configurable
Data capture overhead	Minimum	Raw data stored to disk

Results and Benefits

Reduced Development Complexity

- Removes the need for complex reference models, simplifying development .

Enhanced Performance

- Offloading comparison tasks to scripts improved simulator efficiency by 12%, boosting overall performance.

Faster Debugging

- Visual HTML diff reports reduced debugging time from weeks to days for complex issues.

Scalable and Flexible Solution

- Supports evolving data rates and modulation schemes with configurable granularity options.

Conclusion & Future Work

- Python-based post-processing is a scalable, efficient alternative for SerDes data integrity verification.
- Reduces complexity, accelerates debug, and improves performance.
- Future work: Extend to more protocols, integrate functional coverage, automate scenario generation.

Thank you!

Open for Q&A