RTL Performance Isn't Just a Number - It's a Story

Olivera Stojanovic, Vtool doo, Belgrade, Serbia (oliveras@thevtool.com)

Hagai Arbel, Vtool doo, Belgrade, Serbia (hagai@thevtool.com)

***Abstract*—**Checking ASIC performance differs from functional verification: it's not only about correct outputs but also whether they arrive fast enough, within resource limits, and at expected throughput. Standard performance metrics include latency, throughput, bandwidth, and resource utilization. Best practices treat performance as a first-class metric: define goals, monitor, cover, assert, and link to functional checks. However, identifying the root cause of performance issues remains challenging.

This paper introduces new concepts for extracting and structuring critical data, connecting events in the execution path, adding verification metadata and visualizations. We present measurements with an added dimension, over time, enabling architects to quickly assess expected behavior while guiding designers to focus on critical areas. Furthermore, designers can compare performance data after each design change, understand its impact on all performance parameters, and optimize the design for maximum efficiency.

Keywords—NoC (Network on Chip), SoC (System on Chip) verification, Cogita-PRO, AI (Artificial Intelligence), Big-Data, Performance

I. INTRODUCTION

Current best practice is to treat performance like a first-class verification metric: define clear goals, measure them with dedicated monitors, use cover groups, and enforce limits with assertions. Keep functional and performance checking separated but connected in reports.

Standard DUT performance techniques include following measurements:

1) Latency – time from request to response.
2) Throughput – transactions per cycle/second.
3) Bandwidth – data transferred per unit time.
4) Resource utilization – buffer usage, occupancy, backpressure.
5) Power/performance trade-offs (if modeled).

While these metrics provide a snapshot of performance, they often fail to reveal underlying dynamics or trends over time. This paper introduces a new approach to enhance conventional verification performance measurement, adding temporal and structural insights to better understand DUT behavior. Our approach includes:

1) Number of Outstanding Transactions: Maximum and time-series
2) Duration of transactions over time
3) Normalization Metric: Bytes / Duration per transaction
4) Detect the anomaly in duration- longest transactions in simulation
5) Performance window over time visualization
6) Fairness (e.g., avoiding starvation) and checking arbitration

By combining these extended metrics with visualization and trend analysis, designers and architects gain deeper insights into the DUT's behavior, enabling more informed optimization and debugging decisions.

The results presented in this paper are drawn from Network-on-Chip (NoC) IP.

III. METHODOLOGY

To evaluate DUT performance under realistic operating conditions, specific scenario tests are used that push the DUT to its maximum data-handling capacity. These scenarios are targeted to expose performance bottlenecks and stress resource limits, providing a comprehensive view of system behavior under load.

We leverage the AI-driven debugging platform Cogita-PRO to access, organize, and correlate data required by neural network algorithms, while also providing visualization and interpretation of results. The paper will present trend analysis, and correlation across multiple metrics, supporting precise identification of performance issues.

The primary input to the tool is a log file containing messages marking the start and end of each transaction. For analysis such as bandwidth measurement, the log must include the number of bytes transferred per transaction. Additional transaction fields-such as burst length, burst size, or other relevant signals-can be included to enhance the granularity of the analysis.

This methodology allows designers to:

1) Extract temporal and structural performance metrics.
2) Detect anomalies in transaction duration and outstanding transactions.
3) Visualize performance trends over time.
4) Compare multiple simulation runs to evaluate the effect of design changes on all key performance parameters.

By combining structured data extraction, and visualization, this approach enables a systematic and repeatable performance evaluation workflow that supports informed optimization of the DUT.

Before exploring more complex performance measurements, basic concepts first will be illustrated using a simple AXI interface. We consider AXI requests (AXI_REQ) and AXI responses (AXI_RES). A Route (transaction flow) is created to contain corresponding AXI_REQ and AXI_RES transactions. The tool automatically connects requests and responses using their unique IDs, calculates transaction durations, tracks the number of active transactions, and stores this information in the database.

Figure 1 shows the representation of an AXI Route in the database. Figure 2 presents three graphs derived from this data:

1) Transaction lifecycle graph: Each line represents a single transaction from request to response. This visualization is useful for assessing transfer density and trends in the test, verifying that the DUT is sufficiently stressed, and identifying potential backpressure introduced by design.
2) Outstanding transactions graph: This shows the number of active transactions over time. It provides critical insight for designers, as it directly correlates with buffer sizes in the design and helps guide area-performance optimization decisions.
3) Transaction duration graph: Each bar represents the duration of a single transaction. This measurement highlights the longest transactions in the simulation, enabling the identification of potential performance bottlenecks.

Together, these basic visualizations provide a foundation for understanding DUT behavior and prepare the ground for applying more advanced and time-series.

Id	AXI REQ (utviews-txt-co...	AXI RES (utviews-txt-co...	Total Node Count	Start Timestamp	End Timestamp	Duration	RouteType	RouteSubtype
1	1	1	2	30411811000000	32765601000000	2353790000000	1	1
2	1	1	2	33577971000000	35390181000000	1812210000000	1	1
3	1	1	2	36181721000000	37723141000000	1541420000000	1	1
4	1	1	2	38473021000000	40264401000000	1791380000000	1	1
5	1	1	2	40701831000000	42305741000000	1603910000000	1	1
6	1	1	2	42805661000000	44472061000000	1666400000000	1	1
7	1	1	2	64323051000000	65864471000000	1541420000000	1	1
8	1	1	2	136186551000000	137040581000000	854030000000	1	1
9	1	1	2	137103071000000	137748801000000	645730000000	1	1
10	1	1	2	137311371000000	137977931000000	666560000000	1	1
11	1	1	2	138102911000000	138686151000000	583240000000	1	1
12	1	1	2	138144571000000	139415201000000	1270630000000	1	1
13	1	1	2	138727811000000	139977611000000	1249800000000	1	1

Figure 1AXI Route Table

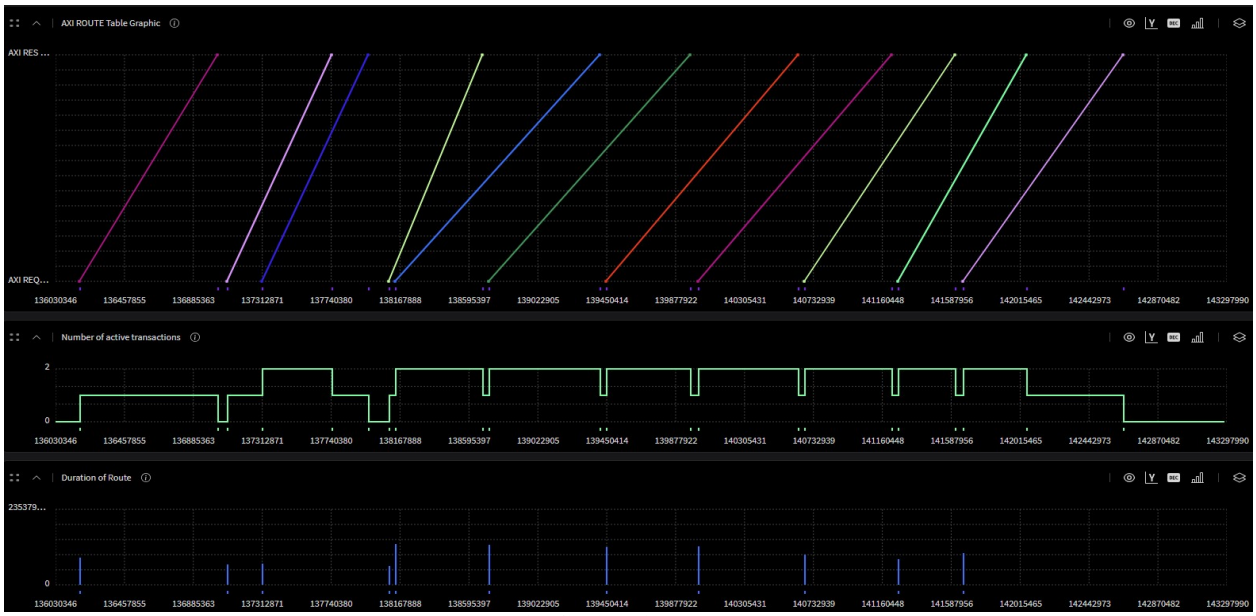

Figure 2 Visualization of AXI Route

IV. RESULTS AND DISCUSSION

The rapid advancement and accessibility of modern technologies open new possibilities for developing new methodologies that provide deeper insight into system behavior and performance. Leveraging these capabilities allows engineers to identify critical areas within simulations that can benefit from design enhancements or configuration adjustments, ultimately leading to more efficient and optimized architecture.

This paper will present flow, proposed methodology and analysis that should be followed while analyzing performance:

1) Quality of performance tests
2) Outstanding Transactions and Performance Characterization
3) Fairness (avoiding starvation)
4) Performance bottlenecks
5) Traffic Efficiency Metrics
6) Sliding window performance

A. Quality of performance tests

The objective of performance test is to have all masters active simultaneously, each generating an equal volume of data traffic evenly distributed across all slaves instances. To validate this, we performed a statistical analysis of traffic distribution for all masters and slaves. In the presented example, there are three masters and four slaves. As shown in Figure 3, all master interfaces exhibit the same distribution all master interfaces exhibit the same distribution Figure 4. After removing the biasing constraint, subsequent test showed a much more even traffic distribution across all slaves instances, aligning with the intended test goals.

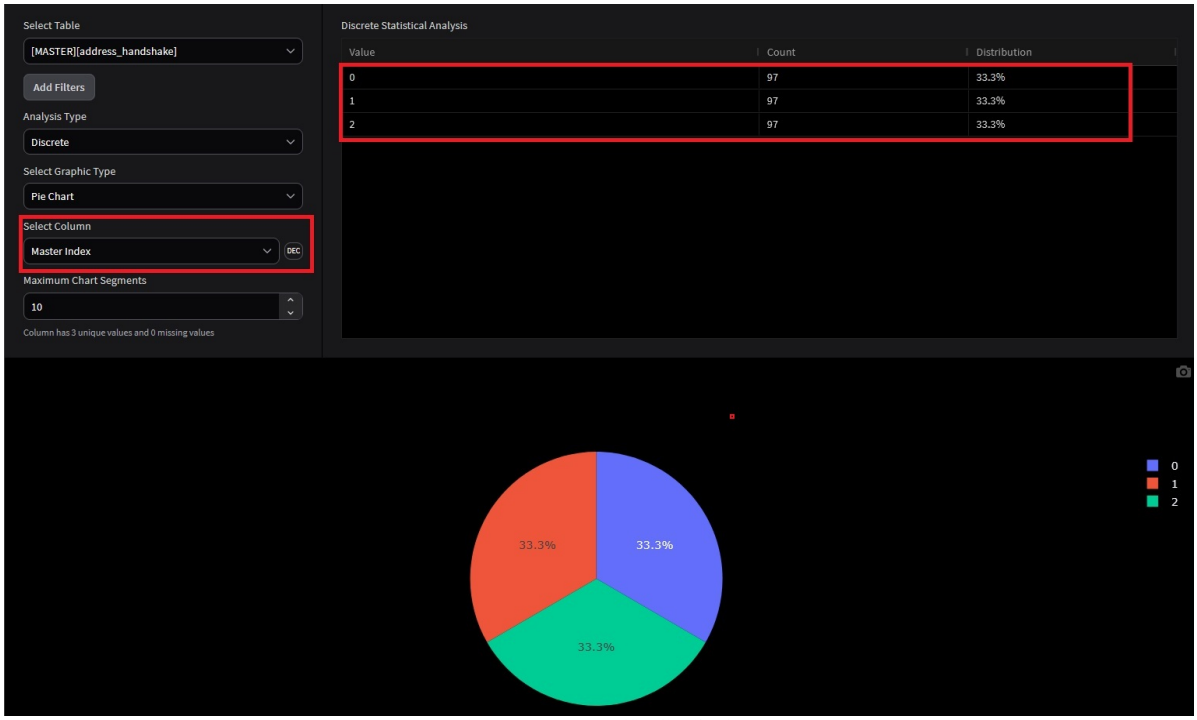

Figure 3 Master traffic distribution

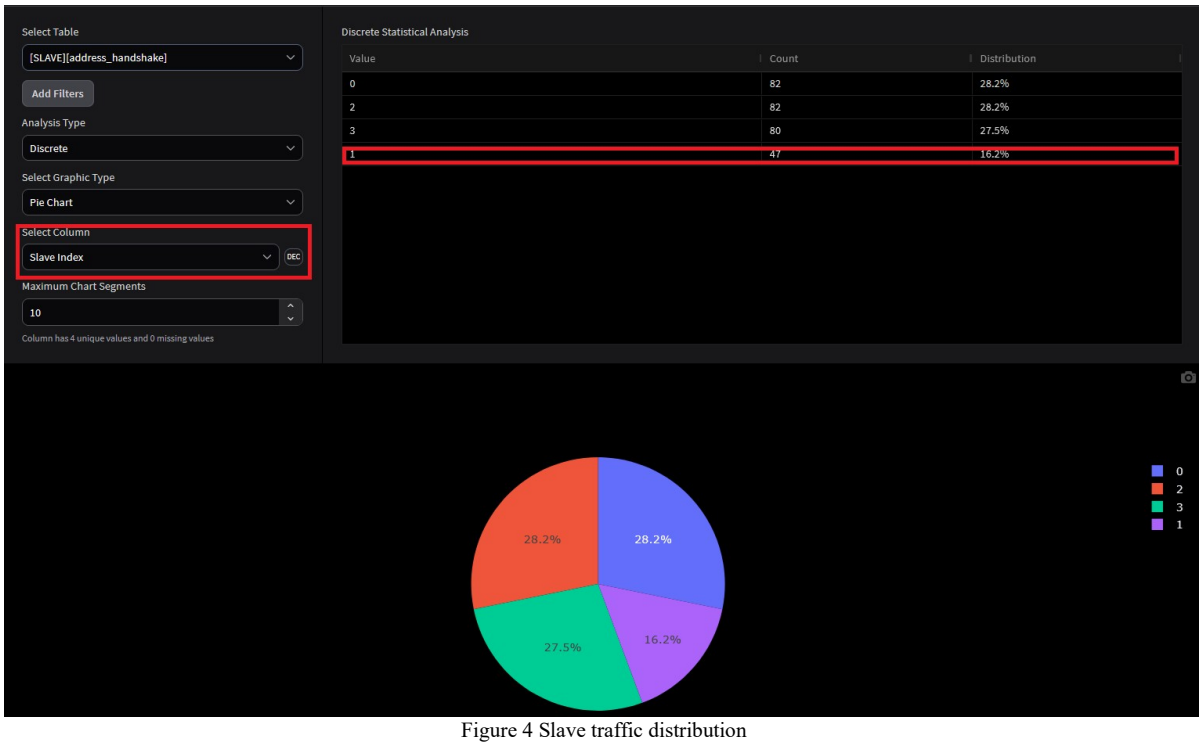

Figure 4 Slave traffic distribution

B. Outstanding Transactions and Performance Characterization

In NoC verification, the ability to control and monitor the number of outstanding transactions is essential for comprehensive stress, coverage, and performance analysis. Proper utilization of outstanding transactions enables the identification of performance bottlenecks, ordering violations, and potential deadlocks, rather than merely saturating the network with random traffic.

Performance in a NoC is strongly influenced by the number of outstanding transactions that each master can issue before receiving responses. Increasing this number can improve throughput by keeping the communication

fabric fully utilized, however, beyond a certain point, excessive outstanding requests can lead to congestion and degraded latency. Therefore, systematic exploration of the relationship between outstanding transactions and transaction duration over time is crucial. By measuring latency as a function of the number of in-flight requests, one can identify the network's saturation point—the threshold beyond which additional concurrency no longer improves, and may even reduce, overall performance.

Standard verification tactics are typically static in nature, focusing on whether the number of outstanding transactions exceeds a predefined maximum value. Coverage metrics are then used to confirm that this limit has been reached during testing, and to report overall performance figures. While such checks are important, they provide only a snapshot of the system's capabilities. To gain a deeper understanding system behavior, it is essential to analyze how the number of outstanding transactions and corresponding performance metrics evolve over time. Observing these dynamics reveals transient effects such as latency accumulation, and buffer utilization trends-factors that are often missed by static verification approaches.

Figure 5 and Figure 6 illustrate the temporal behavior of three masters (M0–M2) within the NoC subsystem. The first figure presents issued transactions per master and the corresponding number of outstanding transactions, while the second figure adds the duration of transactions plotted at their start time.

At the beginning of the simulation, all masters rapidly increase their number of in-flight requests until reaching to configured maximum of outstanding transactions. This is reflected by the steep initial slope in both the issued transaction traces and the outstanding transaction curves. Once this limit is reached, the slope stabilizes, indicating that each new request is issued only after a previous one is completed. This steady-state region corresponds to the saturation phase, where the NoC operates at full concurrency and maximum buffer utilization.

As the simulation progresses, the slopes in the outstanding transaction plot begin to decline, showing that transactions are completed faster than new ones are issued. This transition marks the draining phase of the workload. The duration plot complements this observation: during the saturated phase, transaction durations remain relatively stable, confirming consistent throughput and service time across masters, while greater variation toward the end reflects reduced contention as the system empties its queues.

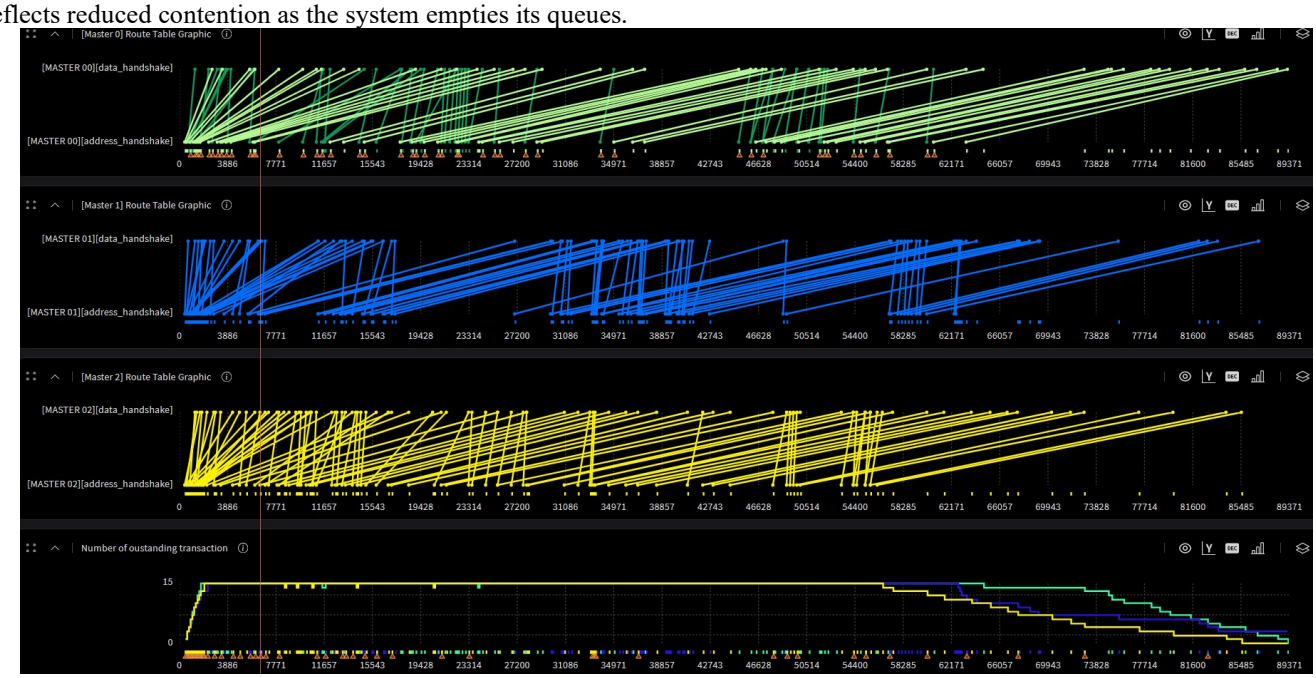

Figure 5 Temporal behavior of three masters

Figure 6 Number of outstanding transactions and duration

The correlation between the rate of change of outstanding transactions and the evolution of transaction duration provides a clear perspective on how the system transitions from full utilization to completion. This analysis also provides valuable insights for chip architects, enabling them to evaluate and optimize different NoC configurations, such as buffer sizing and flow control parameters, to achieve the best trade-off between throughput, latency, and resource utilization.

C. Fairness (e.g., avoiding starvation), round robin or any other algorithm

Fairness in a Network-on-Chip (NoC) ensures that all masters receive equitable access to shared resources, preventing starvation and promoting balanced throughput across the system. Figure 7, the lower plot, the purple graph represents the sequence of masters being granted access to the NoC over time, where the height of each bar corresponds to the master index. The observed switching pattern between different master indices reflects the arbitration behavior of the NoC. A balanced and alternating pattern indicates a fair distribution of bandwidth among masters, while extended sequences of the same master suggest priority-driven access. When correlated with the upper plot showing the number of outstanding transactions, it provides insight into how arbitration policies impact transaction completion and overall fairness in resource allocation.

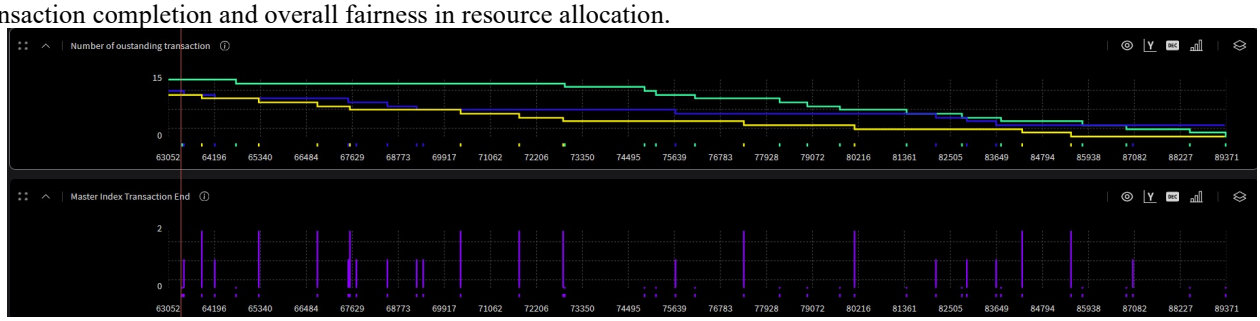

Figure 7 Sequence of masters being granted access

D. Performance bottlenecks

Performance bottlenecks can be effectively identified using anomaly detection algorithms. The algorithm operates on a comprehensive dataset containing detailed information about all routes executed during simulation. By analyzing parameters such as master index, address, burst length, and metadata including transaction duration, the algorithm detects deviations from expected behavior, highlighting potential anomalies that may indicate underlying performance issues.

Figure 8 applies anomaly detection algorithms to identify transactions with abnormal characteristics, where three clear outliers exceed the defined threshold. One anomaly corresponds to transactions with significantly higher duration values, as confirmed in Figure 9 showing the duration evolution over time. The correlation between the anomaly score and prolonged transaction duration indicates that these events represent performance irregularities-likely caused by congestion, arbitration delays, or data dependency stalls. Detecting such anomalies early enables targeted debugging and optimization, helping to improve overall system efficiency and reliability.

The two additional anomalies detected by the tool correspond to transactions with zero duration. Further simulation analysis revealed that a malfunction in the test's timeout mechanism caused the test to terminate prematurely, resulting in the final two transactions completing without receiving responses.

Id	Lin...	Has...	[MA...	MA...	Tot...	Star...	End...	Dur...	Mas...	ope...	add...	Rou...	Rou...
58	6000126b	0	1	1	2	2358000000	561800000...	538220000...	0	WRITE	6000126b	1	1
292	4001d285	0	0	1	1	677150000...	677150000...	0	1	WRITE	4001d285	2	1
293	2000e285	0	0	1	1	693760000...	693760000...	0	1	WRITE	2000e285	2	1
294	400206d9	0	0	1	1	870260000...	870260000...	0	1	WRITE	400206d9	2	1

Figure 8 Anomaly detection

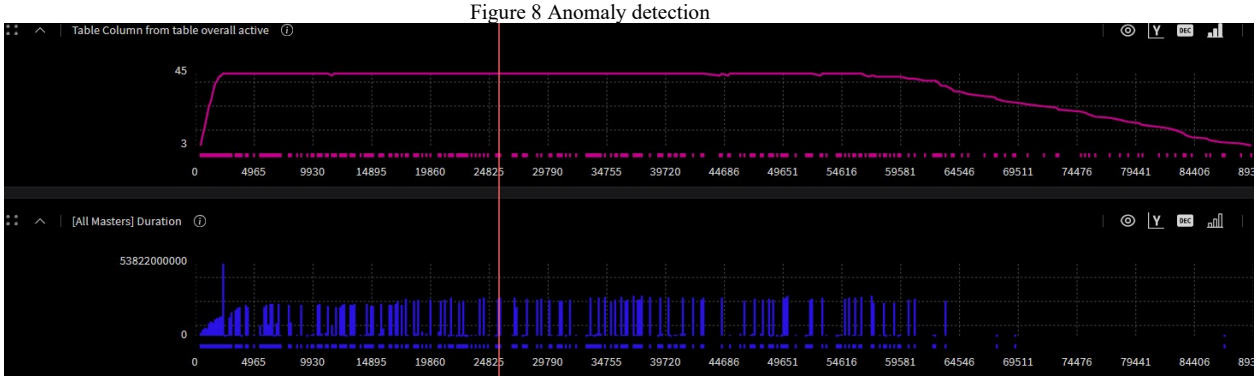

Figure 9 Duration of transactions

E. Traffic Efficiency Metrics

To evaluate the utilization of the interconnect, two complementary metrics can be used: normalized duration per byte and transfer efficiency.

The normalized duration per byte quantifies the average time required to transfer a single byte, incorporating protocol and arbitration overheads. It is computed as the ratio between the total transaction duration and the number of bytes transferred within that transaction. Lower values of this metric indicate higher data throughput and more efficient interconnect utilization, whereas higher values may point to potential performance bottlenecks within the interconnect.

Transfer efficiency represents the proportion of the transaction duration actively used for data transfer, relative to the ideal case of continuous beat transmission. It is calculated as the ratio of the number of transferred bytes to the total transaction duration. Higher efficiency values correspond to better utilization of available bandwidth, while lower values reflect idle cycles or stalls caused by arbitration delays or resource contention.

Together, these metrics provide a quantitative view of how effectively each master utilizes the available interconnect bandwidth, enabling comparison across configurations and traffic patterns.

Figure 10 presents the normalized duration per byte, shown from top to bottom as the aggregate view of all masters, followed by individual plots for each master. The analysis tool automatically identifies the transaction exhibiting the highest normalized duration per byte. This information is critical for designers, as it enables focused inspection of all concurrent transactions at that moment to determine the underlying cause of degraded performance for the identified transaction.

Figure 10 Normalized duration per byte

The histograms Figure 11, Figure 12 and Figure 13 illustrate the efficiency distribution for three masters, showing how effectively each utilizes available bandwidth during transactions. Master 2 demonstrates a slightly broader and higher distribution, indicating it achieves better transfer efficiency compared to master 0 and 1, likely due to more favorable arbitration even with same traffic conditions.

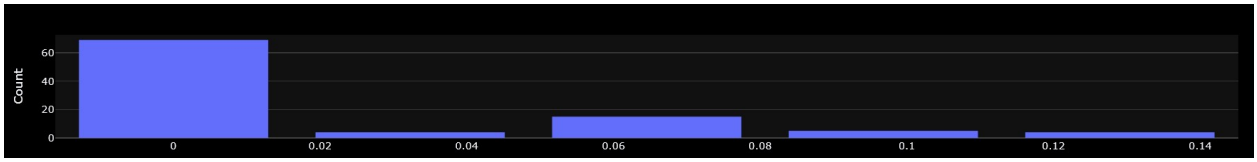

Figure 11 Master 0 efficiency distribution

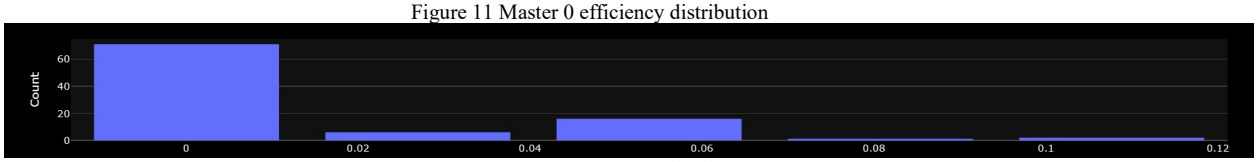

Figure 12Master 1 efficency distribution

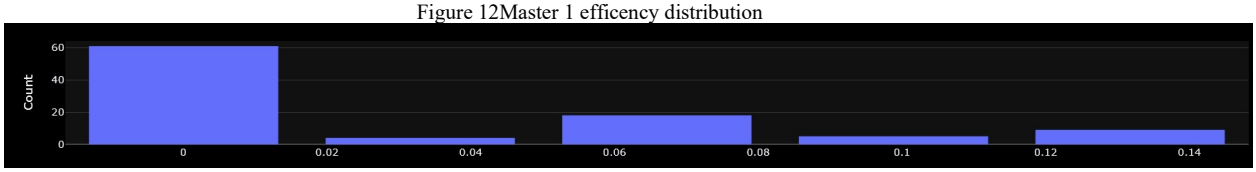

Figure 13 Master 2 efficiency distribution

F. Sliding window performance

Sliding window analysis tracks performance in small, continuous time intervals, exposing short-lived congestion and efficiency variations that are hidden in global averages. Figure 14 presents the evolution of the sliding window metric computed with a step of 1000ns. The upper graph shows variations of the sliding window values, capturing short-term fluctuations in system activity. The lower graph shows the corresponding master activity index, where denser activity regions align with elevated sliding-window values. This correlation highlights how simultaneous master transactions contribute to temporary load increases within the observed window.

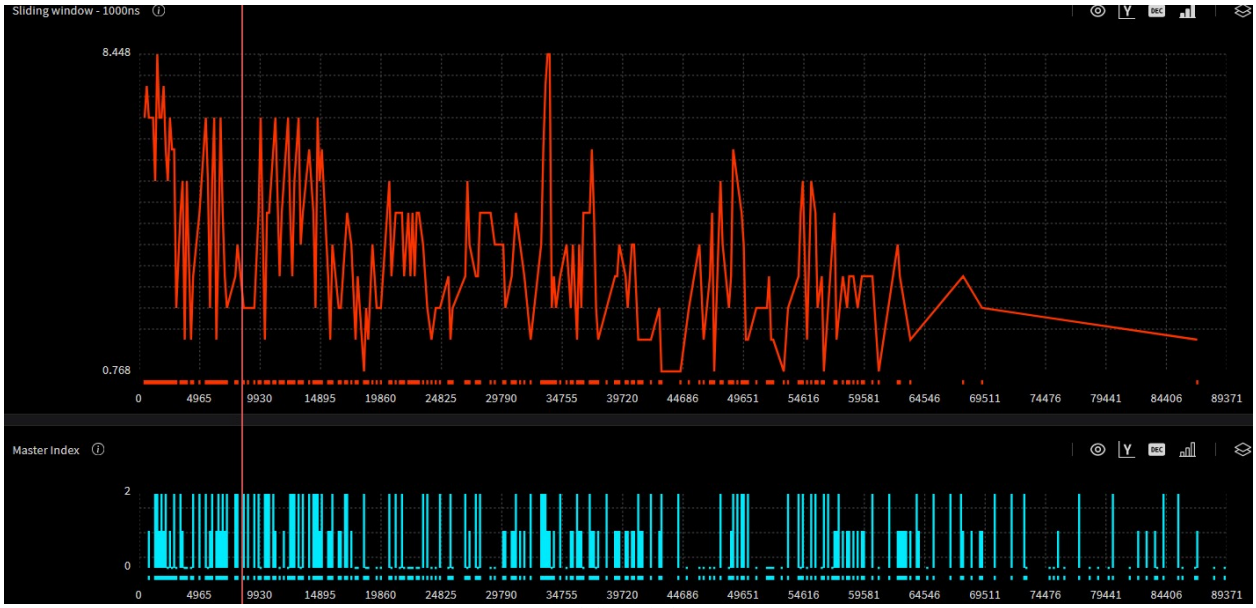

Figure 14 Sliding window

V. CONCLUSION

This paper shows that RTL performance is more than a set of numbers; it reflects the behavior and dynamics of the design over time. By combining standard metrics like latency, throughput, bandwidth, and resource utilization with temporal and structural analyses, designers and architects gain a deeper understanding of how the DUT behaves under different conditions.

Through AI-driven tools such as Cogita-PRO, transaction-level data can be extracted, correlated, and visualized, enabling detection of anomalies, evaluation of trends, and comparison across multiple simulation runs. Basic AXI Route examples demonstrate how lifecycle, outstanding transactions, and transaction duration reveal bottlenecks, backpressure, and opportunities for optimization.

By transforming raw performance data into structured, interpretable insights, this approach allows architects to validate expected behavior and designers to focus on critical areas for improvement. In essence, RTL performance is not just a measurement- it tells the story of how the design truly behaves, guiding informed decisions to achieve efficient, high-performing designs.

REFERENCES

[1] Olivera Stojanovic, Uri Feigin Harnessing the Strength of Statistics and Visualization in Verification" Design and Verification Conference (DVCON), Munich, Germany, 2024