Towards Reliable Systems: A Scalable Approach to AXI4 Transaction Monitoring

Chaoqun Liang*, Thomas Benz†, Alessandro Ottaviano†,
Angelo Garofalo*†, Luca Benini*†, Davide Rossi*
* *Department of Electrical, Electronic, and Information Engineering, University of Bologna*, Italy
† *Integrated Systems Laboratory, ETH Zurich*, Switzerland

***Abstract*—In safety-critical SoC applications such as automotive and aerospace, reliable transaction monitoring is crucial for maintaining system integrity. This paper introduces a drop-in Transaction Monitoring Unit (TMU) for AXI4 subordinate endpoints that detects transaction failures including protocol violations or timeouts and triggers recovery by resetting the affected subordinates.**

Two TMU variants address different constraints: a Tiny-Counter solution for tightly area-constrained systems and a Full-Counter solution for critical subordinates in mixed-criticality SoCs. The Tiny-Counter employs a single counter per outstanding transaction, while the Full-Counter uses multiple counters to track distinct transaction stages, offering finer-grained monitoring and reducing detection latencies by up to hundreds of cycles at roughly 2.5× the area cost. The Full-Counter also provides detailed error logs for performance and bottleneck analysis.

Evaluations at both IP and system levels confirm the TMU's effectiveness and low overhead. In GF12 technology, monitoring 16–32 outstanding transactions occupies 1330–2616 µm^2 for the Tiny-Counter and 3452–6787 µm^2 for the Full-Counter; moderate prescaler steps reduce these figures by 18–39% and 19–32%, respectively, with no loss of functionality. Results from a full-system integration demonstrate the TMU's robust and precise monitoring capabilities in safety-critical SoC environments.

***Index Terms*—AXI4, Transaction Monitor, Reliability, Real-time, Interconnect, Fault Recovery, System-on-chip (SoC)**

I. Introduction

Automotive and aerospace systems demand high reliability and effective fault management in system-on-chip (SoC) designs due to their safety-critical nature. These applications perform complex tasks that require high-performance processing units, advanced safety mechanisms, and robust fault tolerance and recovery strategies to ensure dependable operation under varying conditions. Any failure in the underlying integrated circuits (ICs) within these systems can lead to severe consequences, ranging from equipment malfunctions to complete system outages.

To achieve near-zero Defective Parts Per Million (DPPM) and comply with standards such as ISO 26262 and Automotive Safety Integrity Levels (ASILs) [1], SoC designers must adopt stringent design guidelines that address potential faults and strengthen system resilience. Central to these guidelines is the reliability of the SoC's communication bus, which orchestrates data exchanges between processors, memories, and peripheral devices. A dependable bus prevents errors such as data corruption or protocol violations from propagating, ensuring that high-bandwidth traffic is handled accurately and within required time constraints.

Among the most widely adopted bus protocols for SoCs is the Advanced eXtensible Interface 4 (AXI4) [2], part of the ARM AMBA family. AXI4's support for multiple outstanding transactions increases throughput and lowers latency, making it a strong candidate for complex, high-performance designs. However, this same flexibility introduces complexity that can undermine reliability if not properly safeguarded. Failures can originate from either the manager (e.g., host processor) or subordinate (e.g., memory controller, peripheral device). A subordinate device may fail to respond on time or return incorrect data, leading to deadlocks or data corruption. Conversely, a manager failure can stall the bus entirely, causing missed deadlines and incomplete critical tasks.

To address these challenges, this paper presents a Transaction Monitoring Unit (TMU) IP block for AXI4 interconnect endpoints. Positioned between each subordinate and the AXI4 bus, the TMU detects protocol violations and device timeouts in real-time, triggering fault recovery actions such as issuing a hardware reset to restore malfunctioning subordinates. By isolating faults quickly, the TMU ensures minimal disruption and continued system integrity.

The key contributions of the paper are as follows:

- **A modular Transaction Monitoring Unit (TMU):** Addresses two critical gaps in the state-of-the-art: scalability to monitor multiple outstanding transactions and precision in fault detection.
- **Tiny-Counter solution:** Tracks 16 to 32 outstanding transactions within a compact area ranging from 1330 µm² to 2616 µm², suitable for monitoring typical subordinate devices within heterogeneous, high-performance automotive and aerospace SoCs.
- **Full-Counter solution:** Monitors transactions at the stage level, detecting faults with a latency of just one clock cycle. It also logs performance metrics such as latency and throughput, ideal for critical subordinates requiring detailed analysis.
- **Comprehensive design space exploration:** Demonstrates the TMU's scalability in 12 nm CMOS under four configurations, highlighting the trade-offs between area, monitoring granularity, and performance. Showcases the system's adaptability across different SoC designs.

II. Architecture

The TMU resides between the AXI4 interconnect (manager side) and the subordinate device, continuously monitoring

978-3-9826741-0-0/DATE25/© 2025 EDAA

transactions for protocol violations or timeouts. It is available in two main variants: a *Tiny-Counter (Tc)* for minimal area overhead, and a *Full-Counter (Fc)* for more granular, phase-level analysis (Figs. 1 and 2).

A. ID Optimization and Guard Modules

To handle AXI4's multiple ID lines efficiently, the TMU includes an *AXI ID Remapper* that compacts a wide, sparse ID space into a narrower one. This remapping streamlines transaction tracking. Since AXI4 separates write and read channels, the TMU provides dedicated *Write Guard* and *Read Guard* modules that independently check protocol signals for correctness and timing. A set of software-configurable registers enables or disables the TMU and adjusts parameters such as time budgets, latency statistics, interrupt behavior, and error logging.

B. Normal Operation and Fault Detection

Under normal operation, transactions traverse from the manager (AXI4 interconnect) to the subordinate device without added latency, while the TMU listens in parallel. On detecting a protocol violation or timeout, the TMU severs both request and response paths to prevent error propagation. It then modifies the *slverr* response to the manager, aborts outstanding transactions, and signals an external hardware reset unit [6] to reinitialize the faulty subordinate. Concurrently, it raises an interrupt that prompts the processor to run software-based recovery routines.

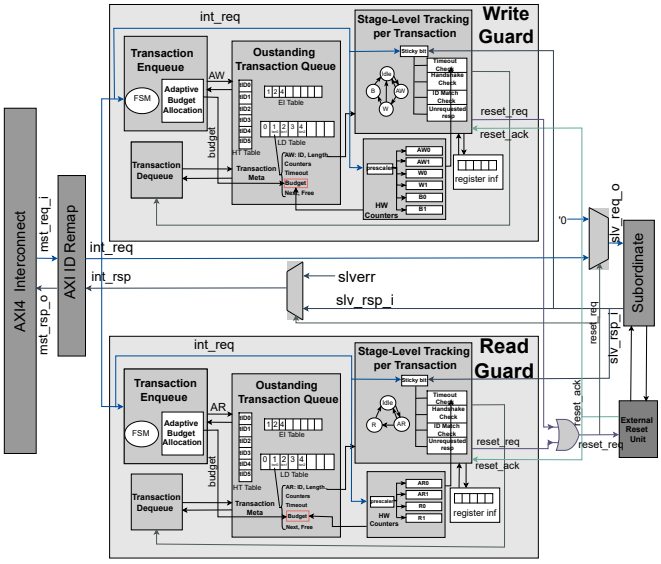


Figure 1: TMU Full-Counter (Fc) Architecture. Multiple counters track distinct phases of each transaction.

C. Outstanding Transaction Tracking

To handle concurrent outstanding transactions from multiple IDs, the TMU enqueues new write or read requests (*aw_valid* or *ar_valid*) into a 2D Outstanding Transaction Table (OTT).

As shown in Fig. 3, the OTT is divided into three linked subtables:

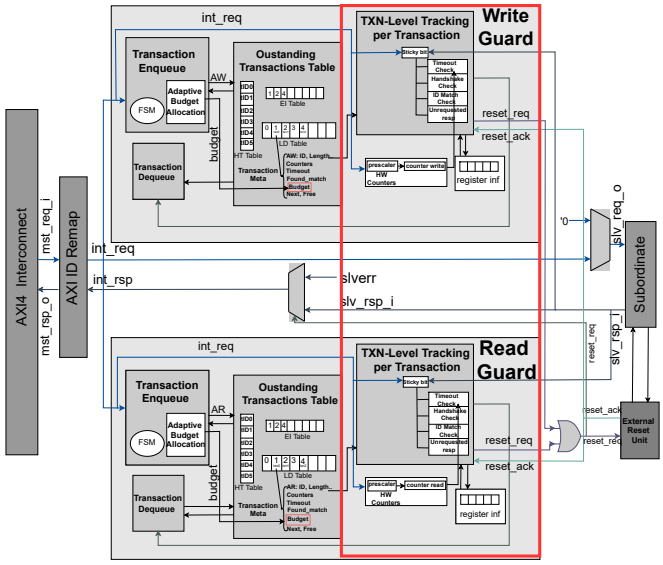


Figure 2: TMU Tiny-Counter (Tc) Architecture. Single counter tracks a transaction. The *red square* highlights logic that differs in the Full-Counter.

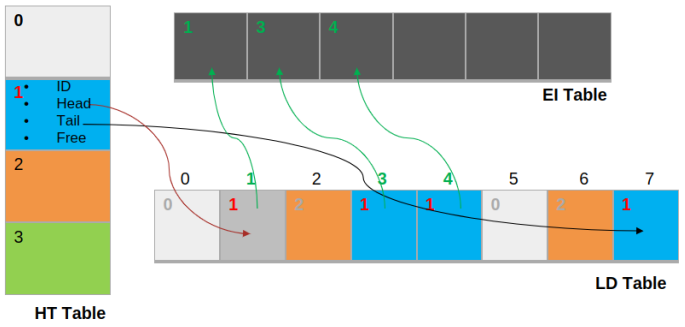


Figure 3: Outstanding Transaction Table (OTT) architecture.

- **ID Head-Tail (HT) Table**: Maintains a FIFO structure for each *tID* to ensure in-order completion of outstanding transactions sharing the same ID as required by AXI4. Each ID's *head* and *tail* pointers link into the LD table.
- **Linked Data (LD) Table**: Stores each outstanding transaction's details, including its *tID*, address, transaction state, budget, latency, timeout status.
- **Enqueue Index (EI) Table**: Enforces the AXI4 requirement that write data on the W channel must follow the same order as the corresponding address transfers on the AW channel. By storing the sequence of AW/AR requests, the EI table ensures that each W beat is correctly associated and presented in the same order. Similarly, for reads, it aligns AR addresses with the R data phase though there is no strict ordering rules for read in AXI4.

D. Tracking Capacity and Resource Management

The TMU's capacity is determined by *MaxUniqID* (the maximum number of unique IDs) and *TxnPerUniqID* (the maximum outstanding transactions per ID). Together, these define *MaxOutstdTxns* as shown in Table II. When the OTT is

saturated, new transaction requests are stalled until an existing transaction completes or is aborted to prevent overflows and preserve system integrity.

Table I: Key Design Parameters

Parameter	Description
MaxUniqIDs	Number of unique Transaction IDs that can be tracked
TxnPerUniqID	Outstanding transactions allowed per ID
MaxOutstdTxns	Total outstanding transactions supported

E. Full-Counter vs. Tiny-Counter Solutions

Full-Counter (Fc): Allocates multiple counters per transaction to track distinct phases separately. For a write transaction, the TMU tracks six key phases (Fig. 4):

1) **Address Handshake:** From *aw_valid* to *aw_ready*.
2) **Data-Phase Entry:** From *aw_ready* to the first *w_valid*.
3) **First Data Transfer Handshake:** From *w_valid* to *w_ready*.
4) **Burst Data Transfer:** From *w_first* to *w_last*.
5) **Response Monitoring:** From *w_last* to *b_valid*, including ID checks and correctness checks.
6) **Response Readiness:** From *b_valid* to *b_ready*.

As illustrated in Fig. 5, a read transaction similarly includes address, data, and response phases tracked by separate counters. Such phase-level timers allow earlier detection of stalls or protocol violations and detailed logging of performance metrics, at the cost of additional hardware.

Tiny-Counter (Tc): The Tc variant illustrated in Fig. 6 assigns a single counter to each outstanding transaction, tracking it from initiation (*aw_valid*/*ar_valid*) until completion (*b_valid*/*r_last*). This approach minimizes hardware overhead but provides only transaction-level granularity: any fault is detected once the overall timeout expires.

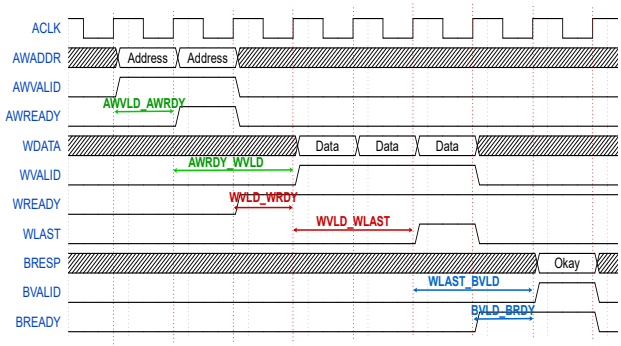


Figure 4: Phase-level monitoring of a write transaction in the Full-Counter (Fc) solution.

F. Adaptive Time-Budgeting Mechanism

To avoid false timeouts in systems with large bursts or burst chaining, the TMU adapts its time budgets based on both burst length and accumulated outstanding traffic in the OTT. For both Fc and Tc solutions, these adaptive budgets are typically divided into two components:

- **Queue Waiting Time:** From the address handshake (*aw_valid* or *ar_valid*) to the first data beat (*w_first* or *r_first*).

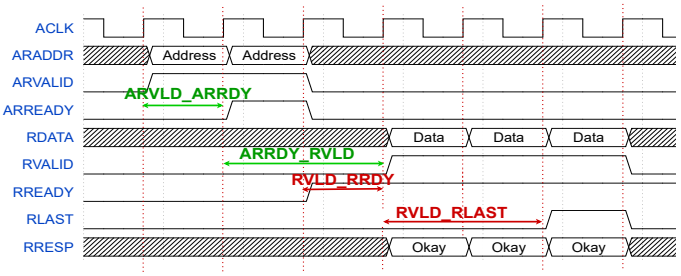


Figure 5: Phase-level monitoring of a read transaction in the Full-Counter (Fc) solution.

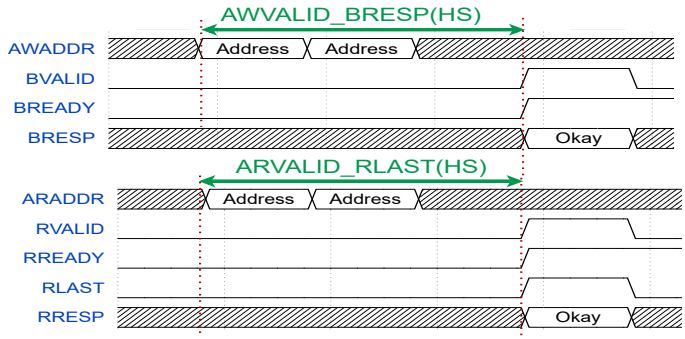


Figure 6: Transaction level monitoring in the Tiny-Counter (Tc) solution.

- **Data Transfer Time:** From *w_first* to *w_last* (writes) or *r_first* to *r_last* (reads).

If any phase extends beyond its allotted time, the TMU flags a timeout.

G. Prescaler and Sticky Bit Mechanisms

To further optimize area, both Tc and Fc can reduce the frequency of counter increments through a *prescaler*. Although this lowers resolution, critical events remain detectable. A *sticky bit* ensures that once a near-timeout condition is observed, it remains recorded even if the counter update is delayed by the prescaler.

H. System Observability and Reliability

By providing real-time tracking of each AXI4 request, the TMU captures latency metrics, identifies bottlenecks, and quickly isolates faulty devices. The Full-Counter solution pinpoints errors down to specific phases, whereas the Tiny-Counter solution offers a cost-effective alternative. In both cases, immediate reset and interrupt capabilities guarantee swift recovery from faults, reinforcing system reliability in safety-critical applications.

III. Evaluation

We evaluate the TMU at both the IP level (synthesized standalone) and the system level (integrated into a real SoC).

A. IP-Level Evaluation

To assess the TMU in isolation, we synthesized it in Global-Foundries' 12nm (GF12) technology under four configurations:

- **Tc**: Tiny-Counter without prescaler or sticky bit.

- **Fc**: Full-Counter without prescaler or sticky bit.
- **Tc+Pre**: Tiny-Counter with prescaler and sticky bit.
- **Fc+Pre**: Full-Counter with prescaler and sticky bit.

1) Outstanding Transaction Setup: We varied the total number of outstanding transactions from 1 to 128 while fixing the number of unique IDs to 4. Within each ID, we explored between 1 and 32 outstanding transactions. This reflects a typical SoC scenario with multiple masters issuing transfers simultaneously, covering both moderate (e.g., 8-32 total transactions) and high concurrency (e.g., up to 128). Each configuration also supports transactions lasting up to 256 clock cycles to accommodate large bursts.

2) Area Results: Figure 7 shows the area results for all four TMU configurations, both with and without a prescaler, illustrating how area scales with the number of outstanding transactions for Tc and Fc. With a prescaler step of 32, the Tiny-Counter plus prescaler (Tc+Pre) consistently consumes the least area, while the Full-Counter without a prescaler (Fc) is the largest. On average, Tc requires about 38% of Fc's area. Across the explored range, prescalers reduce area by 18–39% (Tc) and 19–32% (Fc), trading off timing resolution for hardware savings.

To further examine the impact of prescaler settings, Figures 8(a) and 8(b) plot area versus detection latency for Fc and Tc, respectively, at a fixed capacity of 128 outstanding transactions. Here, the prescaler step ranges from 1 to 128. As expected, larger prescaler values reduce area but increase detection latency, since counter updates occur less frequently and thus can be tracked with a smaller counter width. In this setup, the latency is measured under a scenario where the datapath never asserts a valid signal, effectively modeling a total stall condition.

3) Fault Injection Tests: We validated fault detection and latency by injecting random failures at key AXI transaction stages (Fig. 9). These include:

- AW Stage Error: Missing *aw_ready* acknowledgment
- W Stage Timeout: No valid data received from the master
- W Datapath Error: *w_ready* failure during data transfer
- Data Transfer Error: Issues between *w_first* and *w_last*
- *w_last* to *b_valid* Error
- B Handshake Error: Handshake failure or ID mismatch on the B channel

Phase-specific counters in the Fc solution detect errors earlier and provide detailed performance logging, but at higher area cost. In contrast, the Tc approach offers a single transaction-level counter that reduces hardware overhead but detects errors only after the full transaction time budget.

Overall, the IP-level results confirm that the TMU scales effectively with varying numbers of outstanding transactions and can be tuned via prescalers to meet different area and performance requirements.

B. System-Level Evaluation

To demonstrate full-system reliability, we integrated the TMU into the Cheshire platform [4], a Linux-capable RISC-V CVA6-based SoC, monitoring an RGMII Ethernet peripheral.

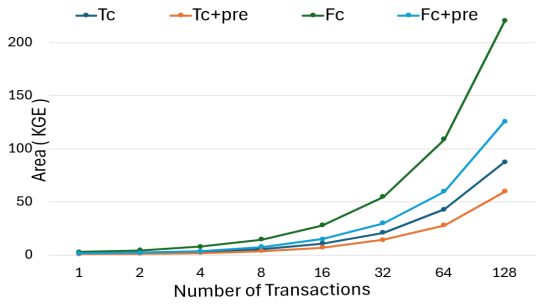
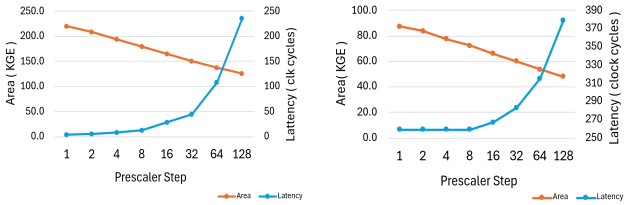


Figure 7: Area comparison of the four configurations (Fc and Tc, with and without prescaler).



(a) Full-Counter Prescaler Exploration (b) Tiny-Counter Prescaler Exploration

Figure 8: Effect of varying prescaler steps on area and detection latency for Fc (a) and Tc (b), at a fixed 128-outstanding-transaction capacity. Larger prescaler values reduce area but increase fault-detection latency.

As shown in Fig. 10, the TMU sits between the AXI crossbar and the Ethernet IP, observing all traffic flowing through Ethernet.

We evaluated a transaction with 250 beats on a 64-bit bus, effectively stressing the Ethernet interface beyond typical packet sizes. The Tiny-Counter (Tc) used a single time budget of 320 clock cycles for the entire transaction, whereas the Full-Counter (Fc) allocated distinct budgets for each phase (e.g., 10 cycles for AW, 250 for W, etc.). Fault injections were identical to those at the IP level. On detecting a timeout or protocol violation, the TMU raises an interrupt and requests an external reset of the Ethernet IP. Upon reset completion, the TMU resumes normal monitoring to ensure continued system stability.

Figure 11 compares detection latencies when injecting errors at the beginning, middle, and end of the transaction. With Tc, detection always occurs after the entire time budget (320

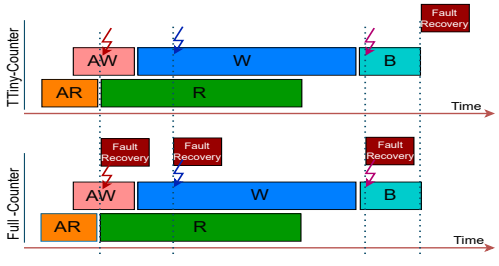


Figure 9: Fault injection setup for IP-level tests.

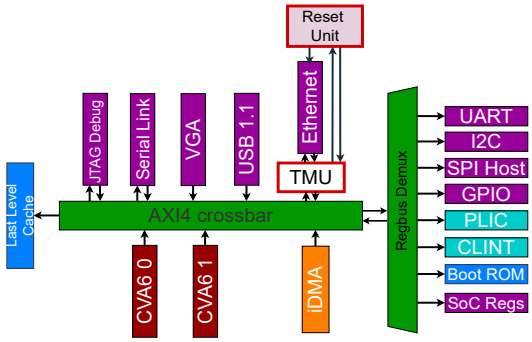


Figure 10: System-level integration: TMU embedded in Cheshire to monitor Ethernet traffic.

cycles). In contrast, Fc signals a fault once the relevant phase times out, providing near-immediate detection if the error occurs early (e.g., during AW).

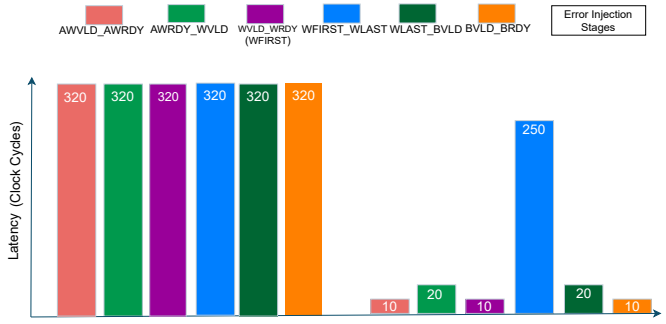


Figure 11: Latency comparison for Ethernet transactions under different fault injections. Fc detects errors earlier by monitoring each phase, while Tc reports errors only at transaction completion.

These system-level findings show that the TMU effectively isolates and recovers from errors with minimal disruption. The Fc variant offers tighter fault detection at increased area cost, whereas Tc provides area savings at the expense of coarser detection granularity.

IV. Related Work

Transaction monitoring in System-on-Chip (SoC) architectures has attracted incresing attention in the contexts of security, performance analysis, fault detection, and reliability. Transaction Monitoring Units (TMUs) are crucial for diagnosing faults in these complex systems, as they monitor transaction events and analyze resource utilization.

Xilinx provides an *AXI Timeout* IP [5] that detects stalls by tracking the time between address and response phases. If a response exceeds a user-defined window, it flags an error and can issue an interrupt, preventing indefinite waits. However, it focuses primarily on prolonged inactivity and lacks phase-level latency metrics, detailed performance logs, and fine-grained protocol checks. As a result, while it prevents severe bus hangs, it does not deliver the comprehensive fault isolation or performance monitoring required in many safety-critical or mixed-criticality SoC environments.

The ARM SP805 Watchdog [6] provides fault detection and system protection by safeguarding SoCs against software malfunctions from unresponsive or runaway processes. It generates interrupts or reset signals upon timeout. Synopsys' Smart Monitor [8] monitors AXI interfaces, capturing AXI bus traffic and computing key performance metrics such as data byte count, throughput, latency, and request/response counts. Similarly, AMD's AXI Performance Monitor [7] measures significant AXI performance metrics, including bus latency for specific manager/subordinate pairs and memory traffic over defined durations.

A security-focused AXI transaction firewall [9] protects subordinate devices from malicious managers by filtering transactions based on operation type (write or read), address range, and required bandwidth against predefined rules. While effective in rejecting unauthorized transactions, it does not monitor transaction timing or detect protocol violations, which are essential for reliable systems.

Ravi [10] introduced a Bus Monitor that captures performance metrics such as transaction count, transfer size, and latency distributions for AXI transactions using hardware counters. Its primary function is to collect behavioral statistics for performance analysis. Lee and Huang [11] developed a Reconfigurable Bus Monitor Tool Suite for on-chip SoC monitoring, offering protocol checking and performance monitoring of bus transactions. Although it verifies compliance issues like address alignment in burst transactions, it overlooks critical rules such as ID mismatches and out-of-order completions and lacks comprehensive performance statistics and fault detection mechanisms.

Kyung [12] presented a Performance Monitoring Unit (PMU) with a Bus Monitor (BM) that observes bus transactions and gathers statistics on address ranges, transfer counts, and read/write latency distributions. Configured to divide transactions into three address ranges, it independently counts bus-transaction events using three performance counter sets. However, this BM does not support multiple outstanding transactions and focuses solely on transaction statistics without addressing error handling or protocol violations.

Chen, Ju, and Huang developed AXIChecker [13], a rule-based, synthesizable protocol checker for AXI transactions within SoC environments. It enforces 44 protocol rules across manager, subordinate, and default subordinate behaviors to ensure compliance with address alignment, burst transactions, and data ordering. While effective for real-time rule-based transaction verification and logging detected violations for debugging, it lacks performance and timing monitoring, limiting its applicability in performance-critical SoC designs.

Tan et al. [14] introduced a method for evaluating the performance of AMBA AXI-based SoC designs, focusing on bandwidth and latency calculations. Edelman and Ardeishar [15] proposed a UVM-based phase-level monitoring approach for simulation-based verification of AXI transactions, capturing transaction phases including address, data, and response within

Table II: Comparison of AXI Transaction Monitors in the Literature

Reference	Target Prot.	HW/SW-Based?	Timing Metrics	Transac. Level	Phase Level	Prot Check	Perf. Metrics	Fault Detection	M.O Supp.*	Scalab.
Xilinx AXI Timeout [5]	AXI	HW	✓	✓	×	×	×	✓	×	×
ARM Watchdog [6]	APB	HW	✓	✓	×	×	×	✓	×	×
AMD Perf. Mon. [7]	AXI	HW	✓	✓	×	×	✓	×	×	×
Synopsys Smart Mon. [8]	AXI	HW	✓	✓	×	×	✓	×	×	×
Lazaro AXI Firewall [9]	AXI	HW	×	✓	×	×	×	×	×	×
Ravi Bus Monitor [10]	AXI	HW	✓	✓	×	×	✓	×	×	×
Lee Bus Monitor [11]	AXI	HW	✓	✓	×	✓	✓	×	×	×
Kyung Perf. Mon. [12]	AXI	HW	✓	✓	×	×	✓	×	×	×
Chen AXIChecker [13]	AXI	HW	×	✓	×	✓	×	×	×	×
Tan Perf. Mon. [14]	AXI	HW	✓	✓	×	×	✓	×	×	×
Edelman Transac. Mon. [15]	AXI	SW	×	×	✓	×	×	×	×	×
This work: Tiny-Counter	AXI	HW	✓	✓	×	✓	✓	✓	✓	✓
This work: Full-Counter	AXI	HW	✓	×	✓	✓	✓	✓	✓	✓

* M.O Supp. denotes Multiple Outstanding Support.

software environments. This method targets pre-silicon verification, ensuring protocol compliance during simulation but does not extend to post-silicon performance monitoring.

Table II summarizes the state-of-the-art works in this field. Our TMU differs by offering comprehensive monitoring at both the transaction and phase levels, supporting multiple outstanding transactions, and providing real-time fault detection and recovery. Notably, the TMU includes advanced features like adaptive time budgeting and detailed performance logging, making it suitable for safety-critical AXI4 systems where in-field fault tolerance is essential. Further, its configurability permits mixing Tiny-Counter and Full-Counter monitors within the same SoC, tailoring overhead and detection granularity to each subordinate's requirements.

V. Conclusion

We presented a versatile Transaction Monitoring Unit (TMU) for AXI4-based SoCs in safety-critical domains such as automotive and aerospace. The TMU provides transaction-level and phase-level monitoring, enabling real-time detection of protocol violations and timeouts, along with rapid fault recovery. Two implementations includign a Tiny-Counter for minimal area and a Full-Counter for more detailed, stage-level monitoring which handles multiple outstanding transactions, adapt time budgets, and log performance metrics. Overall, the TMU delivers a robust, drop-in solution for enhancing reliability, scalability, and performance monitoring in complex AXI4-based SoCs.

Acknowledgment

This work was supported in part by Spoke 1 on Future High-Performance Computing (HPC) of the Italian Research Center on High-Performance Computing, Big Data, and Quantum Computing (ICSC), which received funding from the Ministry of University and Research (MUR) under Mission 4 of the "NextGenerationEU" program.

REFERENCES

[1] S. Im, G. Nam, S. Park and M. Noh, "Advanced safety test solution for automotive SoC based on In-System-Test architecture," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Austin, TX, USA, 2022, pp. 2290-2293, doi: 10.1109/ISCAS48785.2022.9937235.

[2] AMBA AXI Protocol Specification v1.0, ARM, 2003.

[3] T. Benz et al., "A 10-core SoC with 20 Fine-Grain Power Domains for Energy-Proportional Data-Parallel Processing over a Wide Voltage and Temperature Range," in *ESSCIRC 2021 - IEEE 47th Eur. Solid State Circuits Conf. (ESSCIRC)*, Grenoble, France, 2021, pp. 263–266, doi: 10.1109/ESSCIRC53450.2021.9567755.

[4] A. Ottaviano et al., "Cheshire: A Lightweight, Linux-Capable RISC-V Host Platform for Domain-Specific Accelerator Plug-In," *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 70, no. 10, pp. 3777–3781, Oct. 2023, doi: 10.1109/TCSII.2023.3289186.

[5] "AXI Timeout Block," Xilinx.com, 2022. https://docs.xilinx.com/r/en-US/pg080-axi-timeout (accessed Sep. 19, 2024).

[6] "About the Watchdog module (SP805)," Arm.com, Nov. 14, 2003. https://developer.arm.com/documentation/ddi0270/b/introduction/about-the-watchdog-module--sp805- (accessed Sep. 19, 2024).

[7] "AXI Performance Monitor v5.0," Amd.com, Oct. 4, 2017. https://docs.amd.com/v/u/en-US/pg037_axi_perf_mon (accessed Sep. 19, 2024).

[8] "Enhancing Arm SoCs Performance with Smart Monitors," Synopsys.com, Feb. 27, 2023. https://www.synopsys.com/blogs/chip-design/smart-monitors-improve-arm-socs-performance.html (accessed Sep. 19, 2024).

[9] J. Lazaro et al., "Embedded firewall for on-chip bus transactions," *Computers & Electrical Engineering*, vol. 98, 2022, p. 107707, doi: 10.1016/j.compeleceng.2022.107707.

[10] S. Ravi, K. Ezra, and K. H. Mallikarjun, "Design of a Bus Monitor for Performance Analysis of AXI Protocol based SoC Systems," *Int. J. Appl. Eng. Res.*, vol. 9, no. 19, pp. 6313–6324, 2014.

[11] P.-C. Lee and I.-J. Huang, "Reconfigurable Bus Monitor Tool Suite for on-chip SoC for performance and protocol monitoring," in *Proc. 9th Int. Symp. Reconfigurable Communication-Centric Syst.-on-Chip (ReCoSoC)*, Montpellier, France, 2014, pp. 1–6, doi: 10.1109/ReCoSoC.2014.6861341.

[12] H.-m. Kyung et al., "Performance monitor unit design for an AXI-based multi-core SoC platform," in *Proc. 2007 ACM Symp. Appl. Comput. (SAC '07)*, Seoul, Korea, 2007, pp. 1565–1572, doi: 10.1145/1244002.1244336.

[13] C.-H. Chen, J.-C. Ju and I.-J. Huang, "A synthesizable AXI protocol checker for SoC integration," in *Proc. Int. SoC Design Conf. (ISOCC)*, Incheon, Korea, 2010, pp. 103–106, doi: 10.1109/SOCDC.2010.5682961.

[14] T. N. Tan, P. Duong-Ngoc, T. X. Pham and H. Lee, "Novel Performance Evaluation Approach of AMBA AXI-Based SoC Design," in *Proc. 18th Int. SoC Design Conf. (ISOCC)*, Jeju Island, Korea, 2021, pp. 403–404, doi: 10.1109/ISOCC53507.2021.9613920.

[15] R. Edelman and R. Ardeishar, "Monitors, Monitors Everywhere – Who Is Monitoring the Monitors," Mentor Graphics, 2013.