AXI-REALM: A Lightweight and Modular Interconnect Extension for Traffic Regulation and Monitoring of Heterogeneous Real-Time SoCs

Thomas Benz[*x], Alessandro Ottaviano[*x], Robert Balas[*], Angelo Garofalo[*†], Francesco Restuccia[‡], Alessandro Biondi[§], Luca Benini[*†]
[*] *Integrated Systems Laboratory, ETH Zurich*, Switzerland
[†] *Department of Electrical, Electronic, and Information Engineering, University of Bologna*, Italy
[‡] *Department of Computer Science and Engineering, UC San Diego*, San Diego, CA USA
[§] *Department of Excellence in Robotics & AI, Scuola Superiore Sant'Anna*, Pisa, Italy

***Abstract*—The increasing demand for heterogeneous functionality in the automotive industry and the evolution of chip manufacturing processes have led to the transition from federated to integrated critical real-time embedded systems (CRTESs). This leads to higher integration challenges of conventional timing predictability techniques due to access contention on shared resources, which can be resolved by providing system-level observability and controllability in hardware. We focus on the interconnect as a shared resource and propose AXI-REALM, a lightweight, modular, and technology-independent real-time extension to industry-standard AXI4 interconnects, available open-source. AXI-REALM uses a credit-based mechanism to distribute and control the bandwidth in a multi-subordinate system on periodic time windows, proactively prevents denial of service from malicious actors in the system, and tracks each manager's access and interference statistics for optimal budget and period selection. We provide detailed performance and implementation cost assessment in a 12nm node and an end-to-end functional case study implementing AXI-REALM into an open-source Linux-capable RISC-V SoC. In a system with a general-purpose core and a hardware accelerator's DMA engine causing interference on the interconnect, AXI-REALM achieves fair bandwidth distribution among managers, allowing the core to recover 68.2 % of its performance compared to the case without contention. Moreover, near-ideal performance (above 95 %) can be achieved by distributing the available bandwidth in favor of the core, improving the worst-case memory access latency from 264 to below eight cycles. Our approach minimizes buffering compared to other solutions and introduces only 2.45 % area overhead compared to the original SoC.**

***Index Terms*—AMBA AXI, Interconnect, Memory system, Real-time, Automotive, Predictability, Monitoring**

I. Introduction

The growing demand for heterogeneous functionalities with mixed criticality features [1], [2] has led to a *new renaissance* in the design of critical real-time embedded systems (CRTESs) such as modern cars in recent years. *Trusted* (safety- and time-critical) tasks typically include engine, brake, and cruise control, advanced driver assistance system (ADAS), and telematics [3]. These systems comprise single or multi-core general-purpose processors requiring strict certification requirements enforced through timing predictability and composability analysis [4]. *Untrusted* (soft time-critical) computational tasks run infotainment and commodity applications [3]. Untrusted systems are either processors running computational tasks or independent domain-specific accelerators (DSAs) running memory-intensive workloads, such as machine learning (ML) for ADAS [5]. Their applications typically require softer timing bounds and are often not subject to certification processes.

The demand for such heterogeneous functionalities has dramatically raised the complexity of modern real-time cyber-physical systems (CPSs) [6]. The number of electronic control units (ECUs) in cars has grown to as much as 150 per vehicle, but the continuous addition of hardware (HW) has become untenable due to space, weight, power, and cost (SWaP-C) constraints, especially in terms of vehicle cable harnesses [3].

At the same time, the evolution of manufacturing processes has made available more performant systems-on-chip (SoCs) to satisfy such diverse functionality requirements. For this reason, novel real-time architectures in the automotive domain are moving towards merging federated ECUs on the same HW platform [7]. Similarly to integrated modular avionics (IMA) architectures in airborne systems [8], the platform becomes an integrated, heterogeneous mixed criticality system (MCS).

While this solution mitigates the SWaP-C problem, it makes timing predictability and composability analysis more challenging because of the increased interference generated by multiple actors (critical and non-critical multi-core processors (MCPs) clusters, hardware accelerators (HAs), direct memory access (DMA) engines) coexisting on the same platform and contending for shared HW resources. If not considered, the additional contention may introduce unpredictable behavior during the system's execution, causing possible deadline misses for trusted tasks [9], [10]. To preserve the timing behavior of the system under known and predictable bounds, enhancing *observability* and *controllability* of the shared HW resources through temporal isolation becomes a prerequisite [11]. Of particular interest is the interconnect of modern SoCs.

Some existing HW solutions demand intrusive changes to the interconnect, requiring further iterations on the verification process of the intellectual property (IP) [2], [9]. Other approaches are less invasive but lack a cohesive design that addresses both

[x] Both authors contributed equally to this research.

979-8-3503-4859-0/DATE24/© 2024 EDAA

observability and *controllability* of the interconnect [1], [12]. Furthermore, to our best knowledge, all state-of-the-art (SOTA) design proposals limit their cost and functionality assessments to field programmable gate arrays (FPGAs) (Section II).

Contribution: We address the mentioned limitations and propose AXI-REALM, a modular, lightweight, and implementation-agnostic real-time extension to system bus supporting Advanced eXtensible Interface 4 (AXI4), the *de facto* standard for on-chip, non-coherent SoC interconnects. The synthesizable RTL description of AXI-REALM is available free and open-source [1]. We present the following contributions:

1) **HW-driven Traffic Controllability.** AXI-REALM implements a per-manager configurable number of subordinate *regions*, with runtime-configurable address range, budget, and reservation period per *region*. Each manager can be isolated from the system in case of malicious attacks, and a *bus guard* module (Section III) allows privileged access to the unit's configuration registers.
2) **HW-driven Traffic Observability.** A monitoring and regulation unit (M&R unit) tracks per-manager access and interference statistics, such as transaction latency, bandwidth, and interference with each other manager, and eventually performs per-manager budget-aware bandwidth throttling within the period, see Section III-A.
3) **Implementation-agnostic Design.** We integrate AXI-REALM in Cheshire, a Linux-capable RISC-V SoC [13], and characterize it in a 12 nm technology. AXI-REALM incurs an area overhead of only 2.45 % at isofrequency compared to the original design. We can improve a memory-intense benchmark from 0.7 % to 68.2 % of the *single-source* performance with equal budget distribution and *near-ideal* performance when distributing the budget in favor of the core, improving the worst-case memory access latency from over *264* to below *eight* cycles.

II. Related Work

SOTA techniques to improve the real-time capabilities of heterogeneous SoC interconnects traditionally follow two design strategies: adding real-time regulation modules between the managers and the interconnect itself or intrusive customization of existing interconnect architectures. Furthermore, SOTA solutions restrict the design and evaluation of FPGAs or FPGA-SoCs, lacking an open-source hardware platform enabling gate-level characterization in a modern technology node.

Regulation Helper Modules: Several works propose credit-based mechanisms to enforce spatial and temporal upper bounds to non-coherent, on-chip interconnect networks.

Pagani *et al.* and Restuccia *et al.* analyze and address the problem of multiple DSAs competing for bandwidth or causing *denial of service (DoS)* in heterogeneous, AXI4-based FPGA-SoCs and propose three units to mitigate contention. The AXI budgeting unit (ABU) [1] extends the concept of inter-core memory reservation on multi-processor system-on-chips (MPSoCs) to heterogeneous SoCs and proposes counter-based budgets and periods assigned to each manager in the system. The AXI burst equalizer (ABE) [12] tackles unfair arbitration and bandwidth stealing from malicious managers by enforcing a nominal burst size and maximum number of outstanding transactions per manager basis. Finally, the Cut and Forward (C&F) [14] unit moves the burden of completing a transaction from an untrusted manager to the interconnect. A misbehaving manager could cause DoS by reserving write bandwidth and never completing the transaction (stalling). AXI-REALM tackles these challenges by optimizing the design for high performance, adding only one cycle of latency (Section III) and extremely low area overhead (Section IV-A).

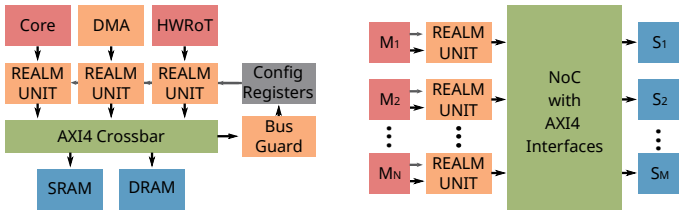
A crucial feature of temporal isolation on shared resources is extracting meaningful information from the functional units during validation, allowing the selection of effective maximum bounds during operation. In [15], Cabo *et al.* propose SafeSU, a minimally invasive statistics unit. SafeSU is limited to tracking inter-core interference in MPSoCs. We extend monitoring capabilities in AXI-REALM to heterogeneous SoCs, tracking average bandwidth, latency, and inter-DSA interference.

Interconnect Customization: This strategy profoundly changes the IPs's inner dataflow. Restuccia *et al.* [9] propose HyperConnect, a custom AXI4-based functional unit block (FUB) for virtualized FPGA-SoCs. While being the closest SOTA to AXI-REALM, HyperConnect cannot prevent DoS from a misbehaving manager stalling the interconnect (Section III-A).

Recently, Jiang *et al.* introduced AXI-IC^{RT} [2], one of the first end-to-end AXI4 microarchitectures tailored for real-time use cases. AXI-IC^{RT} uses the AXI4 user signal to define priorities and proposes a two-layer scheduler algorithm to efficiently select the budget and period for each manager at runtime. Instead, AXI-REALM does not introduce the concept of priority, which may lead to request starvation on low-priority managers. It relies on a credit-based mechanism and a *granular burst splitter* to distribute the bandwidth according to the real-time guarantee of the SoC. While AXI-IC^{RT} evaluates budget reservation strategies, it limits the assessment to managers with equal credit (bandwidth). From an implementation angle, the design strategy followed by AXI-IC^{RT} adds extensive buffering to the microarchitecture to create an observation window for early service of incoming transactions based on priorities. Overall, we observe that both HyperConnect and AXI-IC^{RT} lack monitoring capabilities to track traffic statistics.

In industry, Arm's CoreLink QoS-400 is integrated into modern FPGA-SoCs and controls contention using the QoS signal available in the AXI4 and AXI5 specifications. QoS-400 has several limitations: in a Zynq Ultrascale + FPGA, the authors report more than 30 QoS points that must work coordinately to control the traffic [16]. In the Arm ecosystem, QoS mechanisms are being replaced by memory system resource partitioning and monitoring (MPAM), a promising industry-grade specification with several guidelines to implement memory access regulation mechanisms [17]. MPAM priority partitioning could be applied to AXI-REALM's flexible configuration interface to support bandwidth control from hypervisors integrating MPAM discovery mechanisms.

[1] https://github.com/pulp-platform/axi_rt



(a) (b)

Figure 1. Integration of AXI-REALM in a system with a crossbar and a NoC.

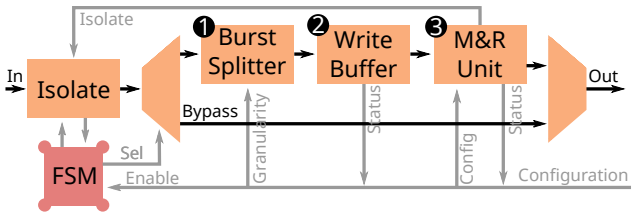
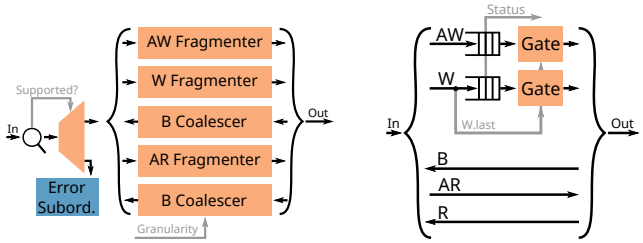


Figure 2. The internal structure of the REALM unit.



(a) (b)

Figure 3. The internal architecture of the granular burst splitter (a) and write transaction buffer (b).

III. Architecture

We design AXI-REALM to be independent of the memory system's architecture, making it compatible with any memory system featuring AXI4 interfaces, from commonly used crossbar-based interconnects [9], [18], [19] to more scalable networks-on-chip (NoCs). Figure 1 shows two example integrations of the real-time regulation and traffic monitoring units (REALM units). We focus on crossbar-based interconnects.

Burst-based interconnect architectures, usually round-robin (RR)-arbitrate transactions on burst granularity [19]. This strategy affects bandwidth distribution fairness by increasing the completion latency of fine-granular transfers in the presence of long bursts issued from DSAs in heterogeneous SoCs. Contrarily to safety- and time-critical tasks executed on processors, DSAs often work independently [1] and can unpredictably cause interference when strict timing bounds are required.

AXI-REALM regulates and monitors the traffic in three stages: ❶ it employs a period- and budget-based scheme to limit manager accesses with predetermined bandwidth on a given time window, a required feature to achieve timing analysis guarantees in real-time systems (M&R unit, Section III-A), ❷ it fragments incoming bursts to a defined granularity, tuned to match the length of the latency-critical transfers (granular burst splitter, Section III-A), and ❸ it forwards write transactions to prevent DoS from malicious managers (write buffer, Section III-A). Compared to existing solutions, budget and period are assigned to a configurable number of subordinate *regions* associated with each manager, providing more flexibility in terms of bandwidth distribution and problem modeling (Section II). Monitoring, modifying, and controlling each manager's traffic at the ingress into the network enables throttling or stalling a stream with minimal congestion and without additional buffering inside the network. AXI-REALM delays in-flight transactions by just one clock cycle. This is negligible, as well-designed AXI4 IPs are insensitive to latency [19]. In the remainder of this section, we provide an in-depth description of AXI-REALM microarchitecture.

A. AXI-REALM Unit

The REALM unit consists of four sub-blocks orchestrated by a finite state machine (FSM), see Figure 2. At the ingress of the module, an isolation block allows the manager to be cut off from the memory system, a beneficial feature in case of misbehaving managers [9]. Isolation is triggered under various conditions, including budget depletion, reconfiguration of intrusive parameters, or user-commanded manager isolation. The isolation block is aware of any outstanding transactions. In the case of user-commanded isolation, it blocks any new transaction while letting outstanding transactions complete.

Before runtime, an operating system (OS) or host hypervisor configures the unit with a period and a budget. Every period, the managers receive *transfer budgets* in bytes, which is spent whenever data is transmitted. If the budget is depleted, the manager is prevented from further executing any more memory accesses until the period expires and the budget is replenished.

Granular Burst Splitter: This unit fragments incoming burst transactions with runtime-configurable granularity. AXI-REALM supports burst fragmentation according to the AXI4 [18] specification. For instance, atomic bursts and *non-modifiable* transactions of length sixteen or smaller cannot be fragmented. Figure 3a shows the unit's microarchitecture. We store the meta-information of a burst transaction, emit the corresponding fragmented transactions, and update and address information. Write responses of the fragmented bursts are coalesced according to the AXI4 specification [18]. Read responses are passed through, except for the *r.last* signal, which is gated according to the length of the original transaction.

The splitting granularity is runtime-configurable from one to 256 beats if the write buffer, see Section III-A, is parametrized large enough or is not present. If a manager only emits single-word transactions, the granular burst splitter can be disabled from the AXI-REALM unit to reduce the area footprint.

Write Buffer: Most interconnect architectures will reserve the bandwidth for an entire write transaction on the *W* channel once the corresponding *AW* is received [19]. A manager device can reserve large amounts of data and stall the interconnect by delaying the corresponding data. A malicious manager can *intentionally* enforce this mechanism to cause DoS of the system, as discussed in [14]. We prevent this behavior by storing the fragmented write burst in a *write buffer*, Figure 3b. The buffer forwards the *AW* request, and the *W* burst once the write data is fully contained within the buffer.

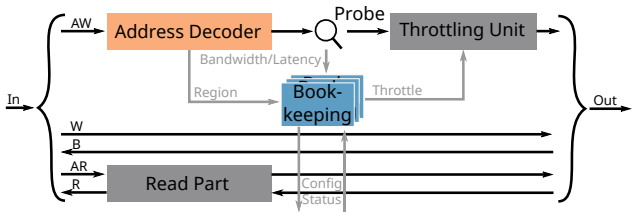


Figure 4. M&R unit: monitors every region and isolates the corresponding manager if the budget is depleted. Read and write transactions are handled equally; the figure only shows the write part exploded.

Read transactions are just passed through, as we assume all of the subordinate devices are returning data in an orderly fashion. The transaction buffer is configured to hold two *AWs* and one fragmented write burst, imposing the largest fragmentation size supported by the granular burst splitter.

Monitoring and Regulation Unit: The M&R unit tracks the transmitted data volume and regulates downstream access with a runtime-configurable period, transfer budget, and address range for each subordinate *region*. The regions are defined independently of the physical routing mechanism of the downstream interconnect. The number of regions is configured at design time. M&R unit's main benefit is increasing the amount of information the SoC [11] exposes to enforce *observability* and *controllability* directly in HW.

Figure 4 shows the write path of the unit exploded, whereas the read path is just implied. The unit analyzes the address of the transaction, determines the corresponding *region*, and subtracts the number of bytes from the region's budget. If at least one of the regions has no budget left, the manager interface is isolated until the budget is replenished, which happens periodically according to the configured reservation period. An optional *throttling unit* can be activated, which limits the number of outstanding transactions to the downstream memory system depending on the remaining budget, modulating backpressure before the budget fully expires.

The M&R unit provides extensive capabilities for online observability. Statistics on the amount of transferred data and elapsed time are measured in relation to the start of the current period, which allows the user to retrieve the region's transfer bandwidth trivially. We further provide average latency measurement capabilities within the bookkeeping units. An increase in average transaction latency indicates contention in the downstream memory system or the subordinate devices. Reading the evolution of the latency from all managers' M&R units and analyzing their statistics provides a full view of the memory system's congestion, extending existing solutions limited to inter-core interference in MPSoCs (Section II).

B. System Integration and Configuration

Depending on the memory system's layout, a shared configuration interface serves one or more REALM units, as shown in Figure 1. In the case of a single AXI4 crossbar integrated at the system level, we use a centralized configuration interface to serve all units. To protect against misbehaving or malicious managers, we propose a *bus guard* unit to restrict unwanted access to the configuration interface. After a system reset, a trusted manager must claim ownership of the configuration

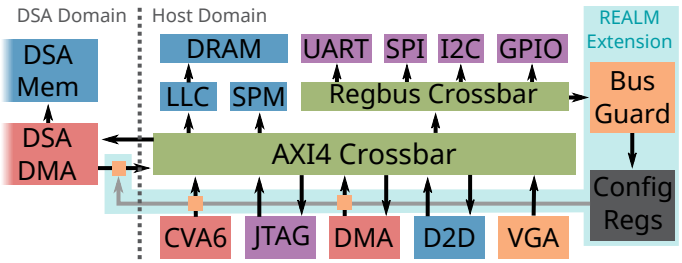


Figure 5. Integration into the Cheshire SoC, ■ denotes the REALM units. Without limiting generality, we show one DSA connected to Cheshire.

space by writing to a *guard register* within the *bus guard*. In the unclaimed state, every access to the configuration space except for the *guard register* returns an error response. Once a manager has claimed an address space within the system, it can perform a *handover* operation to transfer the exclusive read/write ownership to any other manager in the system. The *bus guard* differentiates between managers using their unique transaction ID (TID). If a hardware root of trust (HWRoT) is present in the system, we propose that it claims the configuration register during its boot sequence. Should a more secure approach be required, the AXI-REALM configuration file can directly and exclusively be connected to the HWRoT.

IV. Evaluation

In this section, we provide both in-system and IP-level evaluation of AXI-REALM. In Section IV-A, we integrate the unit in Cheshire, a 64-bit, open-source Linux-capable RISC-V SoC [13]. We synthesize the SoC to evaluate the AXI-REALM area overhead in the system. To assess system-level functional performance, we implement the design on an FPGA and use a representative benchmark from the MiBench automotive suite [20] and the resources available in the RISC-V SoC. In Section IV-B, we perform synthesis on the standalone REALM unit, traversing its design space and deriving an area model and the timing dependency at varying configurations. For gate-level assessment, we use GlobalFoundries' 12 nm node with a 13-metal stack and 7.5-track standard cell library in the typical corner. We synthesize the designs using Synopsys Design Compiler NXT in topological mode to account for place-and-route constraints, congestion, and physical phenomena.

A. In-system Evaluation

We integrate AXI-REALM into Cheshire [13], a 64-bit Linux-capable SoC designed to host AXI4-based DSAs. As shown in Figure 5, a REALM unit is connected to every critical manager used during runtime: the CVA6 core, the SoC-level DMA engine, and the DSAs manager ports. For the evaluation, we configure Cheshire at design time, i.e., before FPGA or ASIC mapping. We select one DSA port connected to an accelerator's DMA engine [21] and add a dedicated scratchpad memory (SPM) to serve as local memory to fully utilize the last-level cache (LLC) as a cache. The REALM unit is parameterized with eight *pending transactions*, a write *buffer depth* of sixteen elements, and two subordinate address regions for each manager. Only one region is used

Table I
AREA DECOMPOSITION OF THE CHESHIRE SOC.

Unit	SoC	CVA6	LLC	Interconnect [a]	*3* RT Units [b]	RT CFG	Peripherals	iDMA	Bootrom	IRQ subsys	Rest
Area [kGE]	3810	1860	1350	206	**83.6**	**9.8**	163	26.3	12.9	11.1	20.5
Area [%]	100.00	48.7	35.5	5.41	**2.19**	**0.26**	4.27	0.69	0.34	0.29	0.54

[a] Without the AXI-REALM extension accounted [b] All 3 units are equally parameterized: 64 b address and data width, a write buffer depth of *16* elements, *eight* outstanding transfers, and *two* available address regions.

Table II
AREA CONTRIBUTIONS OF AXI-REALM'S SUB-BLOCKS AS A FUNCTION OF ITS PARAMETERIZATION. ALL NUMBERS ARE IN GE, AT 1 GHZ USING TYPICAL CONDITIONS.

Parameter	Configuration Register File					REALM unit					
	Per-system	Per-unit	Per-unit & Region			Per-unit				Per-unit & region	
	Bus Guard	Burst config Register	C&S Register	Budget & Period Register	Region Boundary Register	Isolate & Throttle	Burst Splitter	Meta Buffer	Write Buffer	Tracking counters	Region Decoders
Addr Width [ab]	0	0	0	0	20.6	3.5	49.3	38.1	0	0	20.8
Data Width [ab]	0	0	0	0	0	2.7	1.5	0	0	0	0
Num Pending [d]	0	0	0	0	0	9.0	729.4	0	0	0	0
Buffer Depth [d]	0	0	0	0	0	0	0	0	0	0	0
Storage Size [f,g]	0	0	0	0	0	0	0	0	264.4	0	0
Constant [h]	260.6	83.5	24.6	1319.6	0	267.1	4835.0	1309.7	11.4	1928.5	0

[a] In [bit] [b] Evaluated 32 to 64 b [d] Evaluated 2 to 16 elements [f] Product of *Buffer Depth* and *Data Width* [g] Evaluated 256 to 8192 b [h] Base area independent of params

during the evaluation, encompassing the LLC. The REALM units are configured through a dedicated configuration register file connected and mapped into the SoC's address space, see Figure 5. This interface is protected with the *bus guard* introduced in Section III-B. After FPGA mapping, CVA6 claims the configuration registers and initializes the REALM units at an early stage of the boot flow, i.e., *before* runtime operation.

We select *Susan* as a function task for CVA6 from the MiBench suite. *Susan* features the highest memory intensity and is the most representative MiBench automotive benchmark to investigate interference effects in the interconnect bus. The application is executed on top of Linux from the DDR3 dynamic random access memory (DRAM) available on the FPGA as main memory and accessed through Cheshire's LLC. As a baseline, the performance of *Susan* on CVA6 as a *single source* of memory transactions, without the DSA DMA transferring data, highlighted by the *grey dashed line* in Figure 6. In this *single source* case, accesses by CVA6 take at most *eight* cycles to be served by the interconnect, assuming the LLC is hot. During the execution of *Susan* on the core, we program the DSA DMA to perform the worst-case access pattern: *double-buffering* full-length data bursts of *256 beats* between the system's LLC and the DSA's local SPM. Due to the transaction-granular RR arbitration, this leads to the worst-case disturbance and contention, as *every* core access is delayed by *256* cycles. Given this worst-case scenario of *uncontrolled* contention (denoted *without reservation* in Figure 6a), CVA6 requires a minimum of *264* cycles for every memory access, resulting in fewer than 0.7 % of the *single-source* performance.

***Influence of Fragmentation Size**:* To assess the beneficial impact of AXI-REALM under this high contention, we activate the REALM units varying the fragmentation length between *256*, where the unit lets all bursts pass without fragmentation (this corresponds to the *uncontrolled* scenario with high contention *without reservation*), and *one*, where all bursts are decomposed in single-world-granular transfers, restoring fairness from managers launching long bursts. As Figure 6 shows, we can improve the execution time from 0.7 % to 68.2 % of the *single-source* baseline performance at a fragmentation granularity of *one*. We can reduce CVA6's memory access latency from *264* to less than *ten* cycles, only *two* cycles higher than in the *single-source* scenario. One cycle of latency is introduced by the REALM unit, and one can arise from the interference given by the fragmented DMA transfer. In this experiment, we select a very large period and an equal budget for both CVA6 and the DMA to focus on the impact of fragmentation size on traffic fairness.

***Influence of Budget Distribution**:* Moreover, we observe that near-ideal (*dashed line*) core performance can be achieved by reducing the balance between the DMA and the core budget, as depicted in Figure 6b. Decreasing the DMA budget reduces the likelihood of DMA contention, further decreasing CVA6's memory access latency. We configure the fragmentation size to *one* beat, i.e., the optimal fairness condition shown in Figure 6a, with an equal budget for both managers and select a short period of *1000* clock cycles. The DMA's budget is then reduced from 8 kB (denoted by $1/1$) to 1.6 kB ($1/5$) in equal steps.

***In-system AXI-REALM Area Overhead**:* To assess AXI-REALM's resource overhead at the system level, we synthesize Cheshire in the same configuration shown in Figure 5. Overall, the REALM units increase the SoC's area by 83.6 kGE or 2.45 %, as shown in Table I. Introducing AXI-REALM did not reduce Cheshire's maximum operating frequency.

B. IP-level Evaluation

To facilitate area and timing estimates when integrating into different systems, promoting fair comparison with other works, we provide a detailed area model of AXI-REALM in Table II. The data is grouped into two categories: *configuration register file* and the *REALM unit*, grouped into sub-categories *per system*, *per unit*, and *per unit and region*. To estimate the area of an AXI-REALM system, the individual unit's area contributions

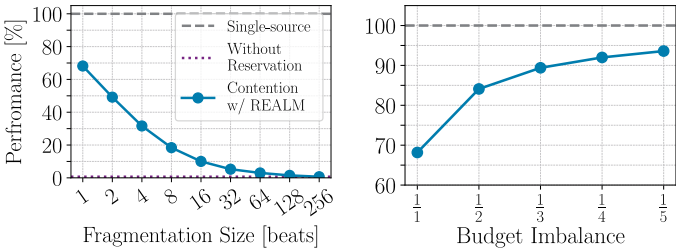


Figure 6. (a) Performance of *Susan* on CVA6 under DSA DMA contention at varying transfer fragmentation in beats. A fragmentation size of *one* provides the most fair scenario and close to ideal performance. *Single-source* denotes the performance without interference. *Without reservation* gives the performance in the case of **uncontrolled** contention. (b) Performance achieved by varying the budget imbalance between the core and the DMA. The fragmentation size is set to one, maximizing fairness under large contention. Traversing the x-axis from left to right, less budget is reserved for the DMA, closing the gap in the core performance under contention compared to the *single-source* scenario.

are multiplied by the parameter value and summed up. The area estimates in Table II are provided at 1 GHz, the target frequency of CVA6 in this node. AXI-REALM can easily achieve clock speeds exceeding 1.5 GHz without adding additional cuts. If an application demands higher frequencies, pipeline stages can be introduced into the REALM unit.

V. Conclusion

This paper presents AXI-REALM, a lightweight, technology-independent interconnect bus extension for heterogeneous real-time SoCs based on AXI4, the de facto standard for non-coherent, on-chip interconnects. We implement AXI-REALM in a SOTA Linux-capable RISC-V SoC, only introducing 2.45 % of area overhead at isofrequency with the original design. AXI-REALM is beneficial to restore bandwidth fairness and limits bandwidth utilization in systems with independent actors issuing high-congestion transactions on a single crossbar, such as HAs in heterogeneous systems. In our experiments with two critical managers contending the system bus (a processor running a representative benchmark from the MiBench suite and a DSA's DMA engine), AXI-REALM can restore the core performance to 68.2 % of the isolated scenario with equal budget distribution across the managers, and near-ideal performance (above 95 %) when distributing the budget in favor of the core, improving the worst-case memory access latency from *264* to below *eight* cycles. We provide an area model of AXI-REALM in a 12 nm node to facilitate rapid exploration and allow fair comparisons. Thanks to AXI-REALM's modularity, use cases beyond real-time embedded computing could be targeted: AXI-REALM could be used in multi-tenant smart NICs [22] to enforce guarantees on shared resource usages. We provide AXI-REALM's RTL free and open-source.

Acknowledgments

We thank Florian Zaruba, Andreas Kurth, Wolfgang Rönninger, Michael Rogenmoser, Paul Scheffler, and Sergio Mazzola, for their valuable contributions to the research project. This work was supported in part through funding from the European High Performance Computing Joint Undertaking (JU) under Framework Partnership Agreement No 800928 and Specific Grant Agreement No: 101036168 (EPI SGA2) and No: 101034126 (The EU Pilot) and by the Spoke 1 on Future HPC of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Mission 4 - Next Generation EU.

References

[1] M. Pagani, E. Rossi, A. Biondi, M. Marinoni, G. Lipari, and G. Buttazzo, "A Bandwidth Reservation Mechanism for AXI-Based Hardware Accelerators on FPGAs," 2019.

[2] I. Gray, Z. Jiang, K. Yang, N. Fisher, N. Audsley, and Z. Dong, "AXI-IC^{RT}: Towards a Real-Time AXI-Interconnect for Highly Integrated SoCs," *IEEE Transactions on Computers*, May 2022.

[3] McKinsey, "The case for an end-to-end automotive software platform," https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/the-case-for-an-end-to-end-automotive-software-platform.

[4] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, "The worst-case execution-time problem—overview of methods and survey of tools," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, may 2008.

[5] Q. Rao and J. Frtunikj, "Deep learning for self-driving cars: Chances and challenges," in *2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS)*, 2018, pp. 35–38.

[6] L. Cuomo, C. Scordino, A. Ottaviano, N. Wistoff, R. Balas, L. Benini, E. Guidieri, and I. M. Savino, "Towards a risc-v open platform for next-generation automotive ecus," in *2023 12th MECO*, 2023, pp. 1–8.

[7] P. Modica, A. Biondi, G. Buttazzo, and A. Patel, "Supporting temporal and spatial isolation in a hypervisor for arm multicore platforms," in *2018 IEEE International Conference on Industrial Technology (ICIT)*, 2018.

[8] C. B. Watkins, "Integrated modular avionics: Managing the allocation of shared intersystem resources," in *25TH ieee/aiaa*, 2006, pp. 1–12.

[9] F. Restuccia, A. Biondi, M. Marinoni, G. Cicero, and G. Buttazzo, "Axi hyperconnect: A predictable, hypervisor-level interconnect for hardware accelerators in fpga soc," in *57th DAC*, 2020, pp. 1–6.

[10] Certification Authorities Software Team (CAST), FAA: Washington, DC, USA, "Position paper–multi-core processors."

[11] F. Rehm, J. Seitter, J. P. Larsson, S. Saidi, G. Stea, R. Zippo, D. Ziegenbein, M. Andreozzi, and A. Hamann, "The Road towards Predictable Automotive High - Performance Platforms," vol. 2021-February. Institute of Electrical and Electronics Engineers Inc., 2 2021.

[12] F. Restuccia, M. Pagani, A. Biondi, M. Marinoni, and G. Buttazzo, "Is your bus arbiter really fair? restoring fairness in axi interconnects for fpga socs," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 5s, oct 2019.

[13] A. Ottaviano, T. Benz, P. Scheffler, and L. Benini, "Cheshire: A lightweight, linux-capable risc-v host platform for domain-specific accelerator plug-in," *IEEE TCAS II: Express Briefs*, pp. 1–1, 2023.

[14] F. Restuccia and R. Kastner, "Cut and forward: Safe and secure communication for fpga system on chips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[15] G. Cabo, F. Bas, R. Lorenzo, D. Trilla, S. Alcaide, M. Moretó, C. Hernández, and J. Abella, "Safesu: an extended statistics unit for multicore timing interference," in *2021 IEEE ETS*, 2021.

[16] A. Serrano Cases, J. M. Reina, J. Abella Ferrer, E. Mezzetti, and F. J. Cazorla Almeida, "Leveraging hardware qos to control contention in the xilinx zynq ultrascale+ mpsoc," *ECRTS*, 2021.

[17] M. Zini, D. Casini, and A. Biondi, "Analyzing arm's MPAM from the perspective of time predictability," *IEEE TC*, 2023.

[18] ARM, "AMBA AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite," 2023, version J.

[19] A. Kurth, W. Rönninger, T. Benz, M. Cavalcante, F. Schuiki, F. Zaruba, and L. Benini, "An open-source platform for high-performance non-coherent on-chip communication," *IEEE TC*, 2022.

[20] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *WWC-4*, 2001, pp. 3–14.

[21] T. Benz, M. Rogenmoser, P. Scheffler, S. Riedel, A. Ottaviano, A. Kurth, T. Hoefler, and L. Benini, "A high-performance, energy-efficient modular DMA engine architecture," 2023.

[22] M. Khalilov, M. Chrapek, S. Shen, A. Vezzu, T. Benz, S. D. Girolamo, T. Schneider, D. D. Sensi, L. Benini, and T. Hoefler, "Osmosis: Enabling multi-tenancy in datacenter smartnics," 2023.