Policy Research Journal
ISSN (E): 3006-7030 ISSN (P): 3006-7022 Volume 4, Issue 3, 2026

DESIGN AND VERIFICATION OF AN FPGA-ORIENTED PRIORITY–AGING QUOTA ARBITER FOR MULTI-MASTER SHARED MEMORY

Muhammad Awais Aslam[*1], Basit Ali[#2], Muhammad Jahan Zaib[#3]

[*1]awaisgrewal286@gmail.com, [#2]basitktk241900@gmail.com,
[#3]mzaib.bee23seecs@student.nust.edu.pk[#3]

[*1,2,3]*National University of Sciences and Technology (NUST), Islamabad, Pakistan*[1,2,3]

DOI: https://doi.org/10.5281/zenodo.21989827

Keywords
arbitration, aging, FPGA, fairness, multi-master memory, priority, quota, shared memory, System Verilog.

Article History
Received: 30 January 2026
Accepted: 15 March 2026
Published: 31 March 2026

Copyright @Author
Corresponding Author: *
Muhammad Awais Aslam

Abstract

Fixed-priority arbitration provides low access latency for critical requesters but may cause excessive waiting or starvation for lower-priority masters, whereas round-robin arbitration improves fairness at the cost of priority awareness. This letter presents a parameterized Priority–Aging–Quota (PAQ) arbiter for three-master access to a shared single-port scratchpad memory. The proposed policy combines base priority with saturating request-age counters and a consecutive-grant quota, while round robin is used only to resolve equal candidates. Urgent requests are selected according to their accumulated age, whereas quota control temporarily prevents a repeatedly granted master from monopolizing the shared resource. The architecture was implemented in System Verilog and integrated with a 256-word, 32-bit shared-memory controller. Verification employed directed tests, constrained-random traffic, assertion-based checking, and a reference-model scoreboard, achieving 100% coverage of the legal and reachable bins defined in the functional coverage model with no detected functional errors. Under an identical request trace, Fixed Priority produced maximum waits of 0, 402, and 568 cycles for the three masters, while Round Robin limited all maximum waits to two cycles. PAQ retained preferential service for the highest-priority master while limiting the observed maximum waits to 2, 4, and 8 cycles. The design was synthesized for a Digilent Nexys A7-100T FPGA, demonstrating the feasibility of PAQ as a configurable compromise between priority preservation and starvation-resistant shared-memory access.

I.INTRODUCTION

The arbitration of concurrent requests to a shared communication or memory resource has long been a fundamental problem in computer architecture. Early schedulers favored simple hardware implementation, with fixed-priority arbitration providing deterministic preference to critical requesters and round-robin arbitration distributing service cyclically among active requesters. Research on fair queueing established that resource allocation could be made more equitable by tracking the service received by competing flows [1], while deficit round robin reduced the implementation complexity of fair scheduling and made it practical for high-speed hardware [2]. McKeown's iSLIP scheduler subsequently demonstrated that iterative round-robin matching could achieve high throughput

and low latency in input-queued switches without requiring prohibitively complex selection logic [3]. As memory bandwidth became a dominant performance constraint, Rixner *et al.* showed that the order in which memory requests are issued strongly affects effective bandwidth and access latency [4]. In parallel, the network-on-chip concept formalized packet-based communication among processors, memories, and accelerators as a scalable alternative to conventional shared interconnects [5]. These foundational studies established the main arbitration objectives that remain relevant today: low latency, high utilization, bounded implementation complexity, and equitable progress among competing requesters.

Subsequent research extended basic arbitration toward quality-of-service, predictability, and interference-aware scheduling. QNoC introduced multiple traffic classes and customized bandwidth allocation to satisfy latency and throughput requirements in systems-on-chip [6], whereas the network-on-chip combined guaranteed-throughput services with a programming and design framework for predictable communication [7]. For shared SDRAM, Predator used predictable access groups and runtime arbitration to provide minimum-bandwidth and maximum-latency guarantees [8]. In general-purpose multicore systems, Stall-Time Fair Memory scheduling addressed unequal slowdown caused by inter-thread memory interference [9], and PAR-BS combined request batching with parallelism-aware ordering to improve both fairness and throughput while supporting thread priorities [10]. Complexity-effective scheduling for many-core accelerator architectures further emphasized that highly parallel request streams require practical hardware-aware scheduling decisions [11]. Thread Cluster Memory scheduling differentiated latency-sensitive and bandwidth-sensitive applications and applied separate policies to the two classes [12]. Later, MemGuard provided bandwidth reservation and temporal isolation for real-time multicore workloads [13], while BLISS used blacklisting to reduce the implementation complexity of application-aware memory scheduling while retaining a balance between performance and fairness [14]. Collectively, these approaches moved arbitration beyond simple requester rotation toward explicit service guarantees, traffic classification, application awareness, and starvation mitigation.

Recent advances have increasingly addressed heterogeneous computing, hardware–software cooperation, and FPGA-based architectural exploration. Priority-aware scheduling under shared-resource contention has used runtime interference and progress information to improve the performance of high-priority applications without completely depriving lower-priority workloads [15]. HD-CPS introduced hardware-assisted communication and dynamically tunable scheduling heuristics for priority-based task execution on shared-memory multicores [16]. At the platform level, open-source FPGA systems have enabled heterogeneous processor and accelerator components to communicate through coherent shared-memory structures, allowing architectural policies to be evaluated in synthesizable hardware rather than only through software simulation [17]. More recently, OpenDRAM presented a modular and extensible soft DDR4 controller for FPGAs, demonstrating that request reordering, bank-parallel scheduling, RTL modularity, timing closure, and on-board validation are all important considerations when translating memory-scheduling research into implementable controllers [18]. These developments indicate a clear transition from isolated arbitration algorithms toward configurable, implementation-aware memory subsystems that must accommodate processors, accelerators, and other requesters with different latency sensitivities and bandwidth demands.

Despite these advances, an important design gap remains between very simple arbiters and sophisticated multicore or DRAM schedulers. Fixed-priority arbitration is compact and preserves requester criticality, but a continuously active high-priority master can cause excessive waiting or starvation for lower-priority masters. Round robin prevents permanent monopolization but treats all requesters uniformly and therefore cannot preserve latency-

critical priority relationships. Many advanced memory schedulers address fairness using application profiling, slowdown estimation, bandwidth reservation, request reordering, or software-visible control, but these mechanisms may be unnecessarily complex for a small FPGA-based shared scratchpad controller. This work therefore proposes a synthesizable Priority–Aging–Quota (PAQ) arbiter that combines four lightweight mechanisms in a frozen decision hierarchy: accumulated waiting time identifies urgent requests, a consecutive-grant quota prevents sustained monopolization when contention exists, base priority preserves criticality among nonurgent requesters, and round robin resolves otherwise equal candidates. The architecture is integrated with a three-master, 32-bit single-port shared-memory controller and evaluated against Fixed Priority and Round Robin using identical request traces. The proposed design targets a practical middle ground: it retains preferential service for a critical master while substantially limiting the observed waiting of lower-priority traffic, without requiring application-level profiling, a complex operating-system scheduler, or an out-of-order DRAM command engine.

II.SYSTEM METHODOLOGY

The proposed research follows a structured methodology comprising architectural modeling, RTL development, functional verification, comparative policy evaluation, and FPGA-oriented implementation. The target system contains three heterogeneous masters competing for access to a common single-port scratchpad memory. Master 0 represents latency-sensitive processor traffic, Master 1 represents burst-oriented accelerator traffic, and Master 2 represents sustained DMA traffic. The methodological distinction of this work is the integration of base priority, request aging, service quota, and Round-Robin tie resolution within a compact and synthesizable arbitration architecture. In contrast to conventional Fixed-Priority arbitration, which can expose lower-priority requesters to excessive waiting, and Round-Robin arbitration, which does not preserve requester criticality, the proposed Priority–Aging–Quota mechanism is designed to retain priority awareness while limiting prolonged service deprivation during contention. The design is parameterized so that Fixed Priority, Round Robin, and PAQ can be evaluated using the same request interfaces, shared memory, traffic conditions, and measurement infrastructure.

IEER
Institute for Excellence in Education & Research

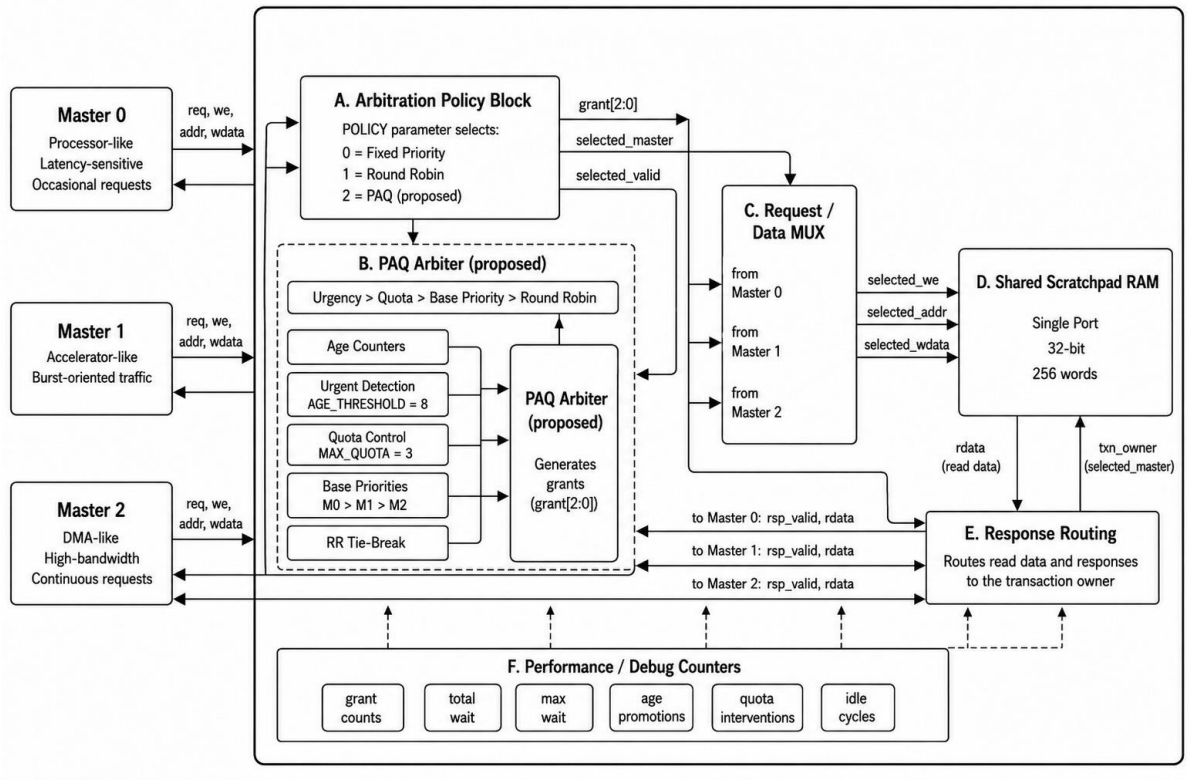


Fig. 1. System methodology of the proposed PAQ-based multi-master shared-memory architecture.

As illustrated in Fig. 1, the three masters provide request, write-enable, address, and write-data signals to the multi-master memory controller. A compile-time POLICY parameter selects one of three arbitration mechanisms: Fixed Priority, Round Robin, or the proposed PAQ policy. The selected arbitration block generates a one-hot grant vector together with selected_master and selected_valid signals. These control the request and data multiplexer, which forwards only the selected master's write-enable, address, and write data to the 256-word, 32-bit single-port scratchpad RAM. Since the memory accepts only one operation in each cycle, the one-hot grant decision prevents simultaneous access by multiple masters. After an accepted transaction, the controller stores the identity of the transaction owner and routes the subsequent completion response and read data back to the appropriate master. In parallel, the controller records grant counts, accumulated waiting cycles, maximum continuous waiting, age-promotion events, quota interventions, and idle cycles for verification and performance analysis.

Within the PAQ policy shown in Fig. 1, each active but unserved request accumulates an age value, while a granted or deasserted request has its age reset. A requester becomes urgent when its age reaches the configured threshold of eight cycles. If one or more urgent requests are present, the arbiter selects the requester with the greatest accumulated age, and Round Robin resolves a tie between equally aged urgent candidates. If no request is urgent, the quota mechanism determines whether the previously selected master has already received three consecutive grants while another requester is waiting. When this condition is satisfied, the previous winner is temporarily removed from the eligible set for the current arbitration cycle. The highest base priority is then selected from the remaining eligible requesters using the ordering M0 greater

than M1 greater than M2, while Round Robin is applied only when equal-priority candidates remain. Consequently, the implemented decision hierarchy is Urgency, Quota, Base Priority, and Round-Robin tie resolution.

The behavioral verification stage was constructed to confirm memory operation, arbitration correctness, response ownership, aging activation, and quota enforcement. Directed tests first exercised read/write routing, the configured base-priority order, the quota boundary, all required quota-handoff conditions, and the age-based promotion of a waiting low-priority requester. These tests were followed by constrained-random traffic with varying requester activity, read/write selection, addresses, and data values. A reference-memory scoreboard checked every completed read transaction, while SystemVerilog assertions monitored one-hot grants, requester eligibility, work-conserving behavior, response timing, response ownership, age updates, urgency generation, and quota enforcement. The resulting PAQ regression is summarized in Fig. 2. It recorded 152, 209, and 121 accepted grants for M0, M1, and M2, respectively; total waiting values of 10, 111, and 300 cycles; and maximum continuous waits of 2, 4, and 8 cycles. The regression also produced seven age promotions for M2, 23 quota interventions, and 97 idle cycles.

```
================ PAQ RESULTS ================
Grants: M0=152 M1=209 M2=121
Total wait: M0=10 M1=111 M2=300
Max wait: M0=2 M1=4 M2=8
Age promotions: M0=0 M1=0 M2=7
Quota interventions=23 Idle cycles=97
Functional coverage (instance) = 100.00%
TEST PASSED: no scoreboard or directed-test errors; inspect transcript for SVA errors.
============================================
```

Fig. 2. PAQ controller verification results under directed and constrained-random traffic.

As shown in Fig. 2, the verification environment completed without scoreboard or directed-test errors and achieved 100% coverage of the legal and reachable bins defined in the controller-level functional coverage model. The nonzero quota-intervention count confirms that repeated service to one master was actively restricted during contention, while the recorded age-promotion events demonstrate that prolonged waiting caused the lowest-priority master to become urgent and receive service. The maximum observed waiting interval of M2 was eight cycles, corresponding to the configured urgency threshold under the evaluated workload. Following functional verification, the same controller framework is used to compare Fixed Priority, Round Robin, and PAQ under identical request traces, after which the verified RTL is synthesized for the Digilent Nexys A7-100T FPGA. This methodology ensures that policy differences are evaluated under common operating conditions and that the proposed PAQ mechanism is assessed in terms of correctness, latency behavior, fairness, hardware feasibility, and implementation cost.

III. RESULTS AND DISCUSSION

The proposed PAQ controller was evaluated through functional verification, identical-trace policy comparison, directed observation of its internal arbitration mechanisms, and FPGA-oriented experimental setup. The verification environment combined directed tests, constrained-random traffic, assertion-based checking, and a reference-memory scoreboard. No scoreboard or directed-test errors were observed, and the defined controller-level functional coverage model reached 100%. The comparative experiment then applied the same cycle-by-cycle request sequence to Fixed Priority, Round Robin, and PAQ so that any differences in grant distribution and waiting behavior

resulted only from the selected arbitration policy. The results show that all three policies remained work-conserving under the evaluated workload, whereas their treatment of priority and fairness differed substantially.

As shown in Fig. 3, Fixed Priority strongly favored the highest-priority master and produced 1445, 1140, and 602 grants for M0, M1, and M2, respectively. Although M0 experienced no waiting, the maximum continuous waits of M1 and M2 increased to 402 and 568 cycles, demonstrating the risk of prolonged service deprivation under persistent high-priority contention. Round Robin restricted the maximum wait to two cycles for every master, but its grant distribution did not preserve the configured priority ordering. In contrast, PAQ generated 1249, 1166, and 772 grants while limiting the observed maximum waits to 2, 4, and 8 cycles for M0, M1, and M2. PAQ therefore retained preferential service for the latency-sensitive M0 requester while preventing the excessive waiting observed under Fixed Priority. All three policies achieved 100% work-conserving throughput, indicating that PAQ improved the priority–fairness trade-off without introducing avoidable idle cycles. The Jain index calculated from the raw grant distribution was 0.9028 for Fixed Priority, 0.9298 for Round Robin, and 0.9631 for PAQ under the applied request trace.

```
================================================================================================================================
THREE-POLICY ARBITRATION COMPARISON
Same cycle-by-cycle request trace applied to all policies
================================================================================================================================
          Policy |   Grants M0/M1/M2 |  Total wait M0/M1/M2 | Max wait M0/M1/M2 | Throughput |     Jain
--------------------------------------------------------------------------------------------------------------------------------
  Fixed Priority |  1445/ 1140/  602 |      0/   963/  2090 |     0/  402/  568 |   100.00% | 0.9028
CSV,Fixed Priority,1445,1140,602,0,963,2090,0,402,568,100.00,0.902835
     Round Robin |   710/ 1052/ 1425 |    735/  1051/  1267 |     2/    2/    2 |   100.00% | 0.9298
CSV,Round Robin,710,1052,1425,735,1051,1267,2,2,2,100.00,0.929760
             PAQ |  1249/ 1166/  772 |    196/   937/  1920 |     2/    4/    8 |   100.00% | 0.9631
CSV,PAQ,1249,1166,772,196,937,1920,2,4,8,100.00,0.963054
================================================================================================================================
PAQ debug at end: age=0/0/0 urgent=000 quota=000 rr=2 last=1 consecutive=0
COMPARISON PASSED: no one-hot, requester, or work-conserving errors.
================================================================================================================================
```

Fig. 3. Three-policy arbitration comparison under an identical cycle-by-cycle request trace.

The directed waveform in Fig. 4 confirms the cycle-level operation of both quota enforcement and age-based promotion. During the quota phase, req_vec = 011 indicates that M0 and M1 are simultaneously active. M0 initially wins because of its higher base priority, and the consecutive-grant counter increases from one to three. After M0 reaches the configured quota of three grants, dbg_quota_block_mask becomes 001, temporarily excluding M0 and allowing M1 to receive grant_vec = 010; the corresponding quota-handoff pulse identifies this transition explicitly. During the aging phase, req_vec = 111 keeps all three masters active. Because M2 has the lowest base priority, its age increases on each unserved cycle. When dbg_age2 reaches seven, the promotion pulse is asserted, and at an age of eight the urgent mask becomes 100. M2 then receives grant_vec = 100, after which its age resets to zero. The waveform therefore confirms that quota prevents short-term monopolization, while aging provides eventual service to a continuously waiting low-priority requester.

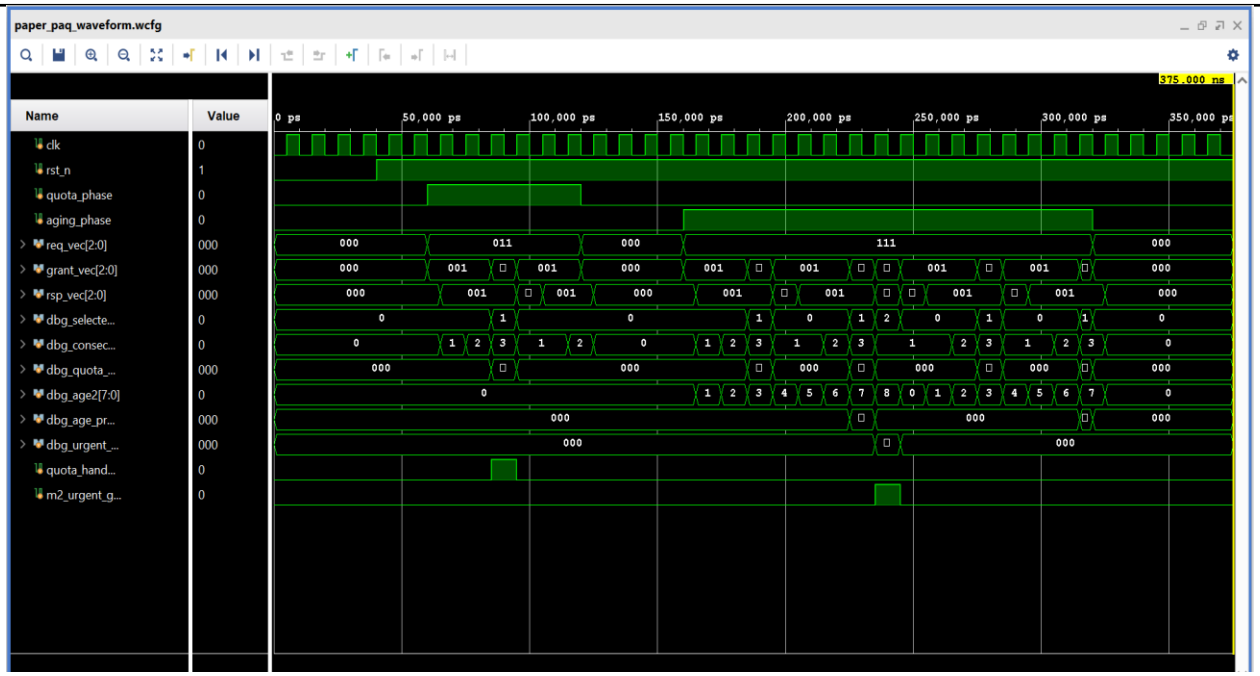


Fig. 4. Directed verification waveform of the proposed PAQ policy showing quota-based service handoff and age-based promotion of M2.

Figure 5 presents the experimental environment used for FPGA-oriented evaluation of the proposed architecture. The setup consists of a Digilent Nexys A7-100T development board connected to a computer running Vivado 2019.1. The Nexys A7 platform provides the XC7A100T FPGA device, a 100-MHz system clock, slide switches, LEDs, push buttons, and an eight-digit seven-segment display, enabling the controller to be synthesized and prepared for board-level observation. In the developed hardware top, switch inputs select the traffic workload and displayed performance statistic, while LEDs indicate request, grant, urgency, quota-blocking, initialization, response, and protocol-error states. The experimental configuration supports the collection of FPGA utilization, timing, power, and hardware-behavior results while maintaining the same PAQ parameters used during RTL verification.

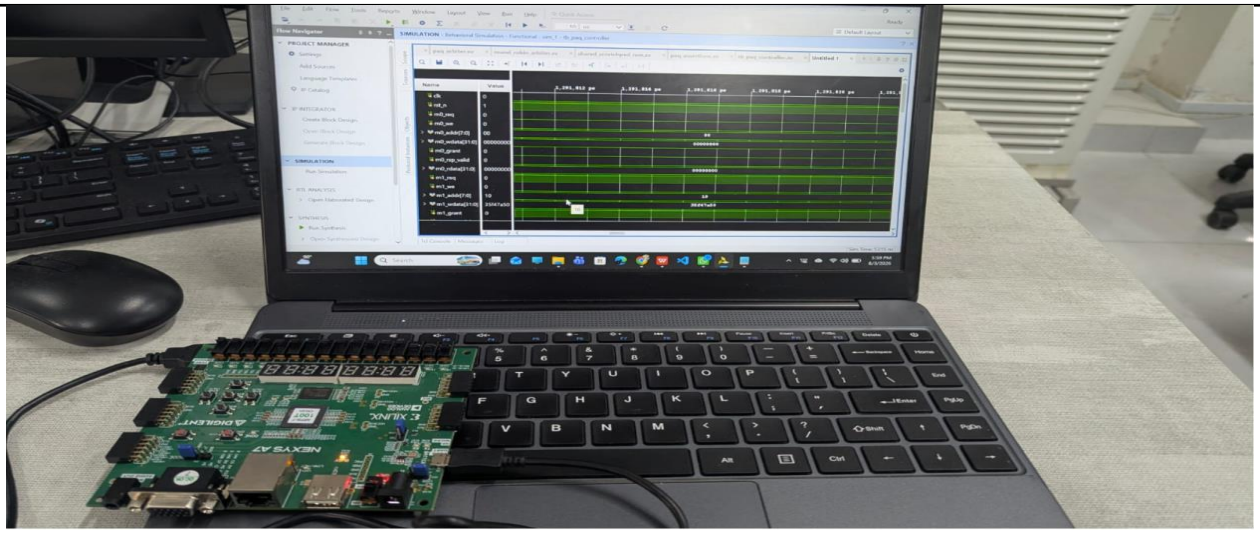


Fig. 5. FPGA-oriented development setup for the PAQ controller using the Digilent Nexys A7-100T platform and Vivado 2019.1.

Overall, the results demonstrate that the proposed PAQ mechanism provides a practical compromise between strict priority and equal cyclic service. Fixed Priority offers the strongest preference to M0 but exposes lower-priority masters to very large waiting intervals, whereas Round Robin minimizes and equalizes waiting at the cost of ignoring requester criticality. PAQ preserves higher service preference for M0 while restricting the observed maximum waiting intervals of M1 and M2 through quota and aging. The directed waveform verifies that these improvements arise from the intended internal mechanisms rather than from changes in the offered workload, and the experimental platform establishes a path toward physical FPGA evaluation. Consequently, the proposed architecture is suitable for heterogeneous shared-memory systems that require priority awareness, starvation resistance, work-conserving operation, and comparatively low implementation complexity.

IV. CONCLUSION

This work presented the design, verification, and FPGA-oriented evaluation of a parameterized Priority–Aging–Quota arbiter for three-master access to a shared single-port memory. The proposed PAQ policy combines base-priority preservation with age-based urgency, consecutive-grant quota control, and Round-Robin tie resolution, thereby addressing the excessive waiting associated with Fixed Priority while avoiding the complete loss of priority awareness inherent in conventional Round Robin. The SystemVerilog implementation was validated using directed tests, constrained-random traffic, assertions, and a reference-memory scoreboard, achieving 100% coverage of the legal and reachable bins defined in the controller-level functional coverage model without detected functional errors. Under an identical request trace, all policies maintained 100% work-conserving throughput; however, Fixed Priority produced maximum waits of 0, 402, and 568 cycles for M0, M1, and M2, whereas PAQ limited the observed maximum waits to 2, 4, and 8 cycles while retaining preferential service for the highest-priority master. Directed waveforms further confirmed correct quota-based handoff and age-triggered promotion of the lowest-priority requester. These results demonstrate that PAQ provides a practical and comparatively lightweight trade-off between latency criticality and starvation resistance for heterogeneous shared-memory systems, while its synthesizable and parameterized structure supports implementation and further evaluation on the Digilent Nexys A7-100T FPGA platform.

REFERENCES

Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. ACM SIGCOMM*, 1989, pp. 1–12, doi: 10.1145/75246.75248. (DBLP)

M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Trans. Netw.*, vol. 4, no. 3, pp. 375–385, Jun. 1996, doi: 10.1109/90.502236. (Pascal and Francis)

N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Netw.*, vol. 7, no. 2, pp. 188–201, Apr. 1999, doi: 10.1109/90.769767. (AbleSci)

S. Rixner, W. J. Dally, U. J. Kapasi, P. R. Mattson, and J. D. Owens, "Memory access scheduling," in *Proc. 27th Int. Symp. Computer Architecture (ISCA)*, 2000, pp. 128–138, doi: 10.1109/ISCA.2000.854384. (DBLP)

W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. 38th Design Automation Conf. (DAC)*, 2001, pp. 684–689, doi: 10.1145/378239.379048. (DOI)

E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *J. Syst. Archit.*, vol. 50, nos. 2–3, pp. 105–128, Feb. 2004, doi: 10.1016/j.sysarc.2003.07.004. (ScienceDirect)

K. Goossens, J. Dielissen, and A. Rădulescu, "Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Design Test Comput.*, vol. 22, no. 5, pp. 414–421, Sep.–Oct. 2005, doi: 10.1109/MDT.2005.99. (Eindhoven Research Portal)

Akesson, K. Goossens, and M. Ringhofer, "Predator: A predictable SDRAM memory controller," in *Proc. 5th IEEE/ACM/IFIP Int. Conf. Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2007, pp. 251–256, doi: 10.1145/1289816.1289877. (Eindhoven Research Portal)

O. Mutlu and T. Moscibroda, "Stall-time fair memory access scheduling for chip multiprocessors," in *Proc. 40th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, 2007, pp. 146–160, doi: 10.1109/MICRO.2007.21. (Microsoft)

O. Mutlu and T. Moscibroda, "Parallelism-aware batch scheduling: Enhancing both performance and fairness of shared DRAM systems," in *Proc. 35th Annu. Int. Symp. Computer Architecture (ISCA)*, 2008, pp. 63–74, doi: 10.1145/1394608.1382128. (Microsoft)

G. L. Yuan, A. Bakhoda, and T. M. Aamodt, "Complexity effective memory access scheduling for many-core accelerator architectures," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, 2009, pp. 34–44, doi: 10.1145/1669112.1669119. (ResearchGate)

Y. Kim, M. Papamichael, O. Mutlu, and M. Harchol-Balter, "Thread cluster memory scheduling: Exploiting differences in memory access behavior," in *Proc. 43rd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, 2010, pp. 65–76, doi: 10.1109/MICRO.2010.51. (DBLP)

H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha, "MemGuard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms," in *Proc. IEEE 19th Real-Time Embedded Technology Applications Symp. (RTAS)*, 2013, pp. 55–64, doi: 10.1109/RTAS.2013.6531079. (Illinois Experts)

L. Subramanian, D. Lee, V. Seshadri, H. Rastogi, and O. Mutlu, "BLISS: Balancing performance, fairness and complexity in memory access scheduling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 3071–3087, Oct. 2016, doi: 10.1109/TPDS.2016.2526003. (DBLP)

S. Kundan and I. Anagnostopoulos, "Priority-aware scheduling under shared-resource contention on chip multicore processors," in *Proc. IEEE Int. Symp. Circuits Systems (ISCAS)*, 2021, pp. 1–5, doi: 10.1109/ISCAS51556.2021.9401337. (ResearchGate)

M. Shan and O. Khan, "HD-CPS: Hardware-assisted drift-aware concurrent priority scheduler for shared memory multicores," in *Proc. IEEE Int. Symp. High-Performance Computer Architecture (HPCA)*, 2022, pp. 528–542, doi: 10.1109/HPCA53966.2022.00046. (ResearchGate)

R. Tornero, D. Rodríguez, J. M. Martínez, and J. Flich, "An open-source FPGA platform for shared-memory heterogeneous many-core architecture exploration," in *Proc. 38th Conf. Design Circuits Integrated Systems (DCIS)*, 2023, pp. 1–6, doi: 10.1109/DCIS58620.2023.10335973. (Panorama)

Abbasi, D. Germchi, A. Katani, M. Hassan, and R. Pellizzoni, "OpenDRAM: A modular, high-performance soft memory controller for DDR4 DRAM," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 18, no. 4, Art. no. 56, pp. 1–27, Dec. 2025, doi: 10.1145/3772724. (DOI)

IEER
Institute for Excellence in Education & Research