FPGA Design and Performance Evaluation of a Bus Arbitrator Based on Dynamic Priority Adjustment

Yibo Wang *

School of Electronic and Information Engineering, Tongji University, Shanghai, 201800, China

* Corresponding Author Email: 2452312@tongji.edu.cn

Abstract. With the continuous advancement in system-on-chip (SoC) integration, balancing real-time responsiveness and fairness in bus access among multiple master devices has become essential for improving the overall performance and energy efficiency of complex SoCs. This study designs and implements a dynamic priority bus arbiter based on a multi-parameter weighting scheme that incorporates four factors: priority, waiting time, resource demand, and an emergency flag.This approach enables arbitration decisions that achieve high real-time performance, fairness, and low resource overhead. By establishing a test platform on Vivado for synthesis and simulation, the experimental results demonstrate that the proposed arbiter reduces the average grant delay to 1.82 ns and responds to urgent requests within 1.34 ns at a clock frequency of 500 MHz. While effectively controlling the logic resource utilization and power consumption, the arbiter ensures both practicality and superior fairness compared with conventional static arbitration strategies. This research is of great significance in promoting the application of next-generation intelligent computing chips in real-time-sensitive scenarios such as autonomous driving and edge computing.

Keywords: Dynamic priority, Network-on-chip, Bus contention, Real-time systems.

1. Introduction

As the core component of modern electronic systems, system-on-chip (SoC) integrates multiple processor cores, memory, peripheral interfaces, and other functional IP cores on a single chip, improving the integration and performance of the system. However, with the rapid development of semiconductor technology and the continuous improvement in the complexity of integrated circuit design, the number of main devices inside the system has surged, and the demand for concurrent access to the bus has also increased sharply, which has led to the gradual emergence of resource competition. The Arbiter plays a crucial role in hardware systems by facilitating the fair allocation of resources among competing devices [1].

Arbiters receive multiple requests as inputs and determine access prioritization, typically through the generation of a one-hot signal output in Fig. 1[1]. They directly impact system efficiency and real-time performance, and are widely used in fields such as autonomous driving and industrial production [2].

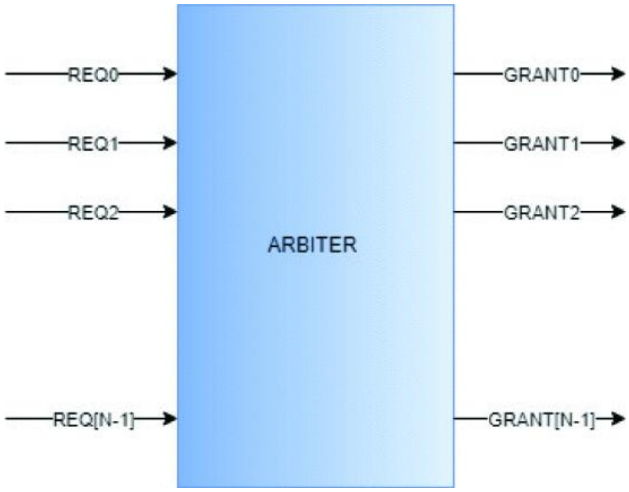

Fig. 1 A general representation of an arbiter

While static arbitration algorithms exhibit inherent limitations when handling varying system states and diverse request profiles. Dynamic arbitrators were developed specifically to address these shortcomings. This type of arbitrator often uses a centralized decision-making mechanism, where all incoming requests are evaluated at the same time to determine access priority, enabling it to dynamically adjust access allocation based on changes in system state and request characteristics [1]. In recent years, the continuous advancement of multi-core processors, chip-level neural networks [2], and High-Performance Computing has deepened research into dynamic arbitration. Design philosophies have shifted from simple priority adjustments toward intelligent mechanisms, including machine learning and dynamic programming [3][4].

For example, Zhu Yaqi et al. designed a DMA data stream arbiter. It used a bit-map algorithm and eliminated priority comparison circuits [5]. Li Xin implemented a PUF-based arbiter. Its principle is simple and security is excellent under bias control [6][7]. A multi-priority matrix arbiter achieved higher fairness. It dynamically updated priorities in real time [8].

This paper systematically investigates the design methodologies, performance evaluation, and optimization strategies for dynamic arbitrators in digital circuits. By analyzing and comparing existing representative dynamic arbitration algorithms and integrating specific application scenarios such as network on-chip and multi-core processors, a dynamic arbitrator architecture featuring high throughput, low latency, and strong adaptability was designed. Its effectiveness is validated through simulation and hardware implementation.

2. The Theoretical Foundations of Arbitration Strategy

2.1. Traditional Arbitration Strategies

Traditional arbitration strategies primarily include static arbitration algorithms such as fixed priority (FP) and round-robin priority (RR). Figures 2 and 3 illustrate their algorithmic structures [8]. Fixed priority algorithms are simple to implement and offer fast response times, but they can easily lead to low-priority devices being unable to gain bus access for extended periods, resulting in "starvation"[9]. Although RR arbiter offers good fairness and is more advantageous for applications requiring higher clock frequencies and smaller chip areas [10]—such as the 8-bit 4x4 crossbar switch designed for multiprocessor OSAs using a polling arbitration algorithm in [11]—it cannot accommodate the varying real-time and bandwidth demands of different devices, performing poorly in imbalanced load scenarios.

Additionally, there also exist some random allocation algorithms, such as Lottery arbitration. It can balance bandwidth distribution to some extent. But it has significant randomness and complex hardware implementation [12]. Traditional strategies are generally simple to implement. They are widely adopted. For instance, IP core traffic management through arbitration improved communication efficiency [13].

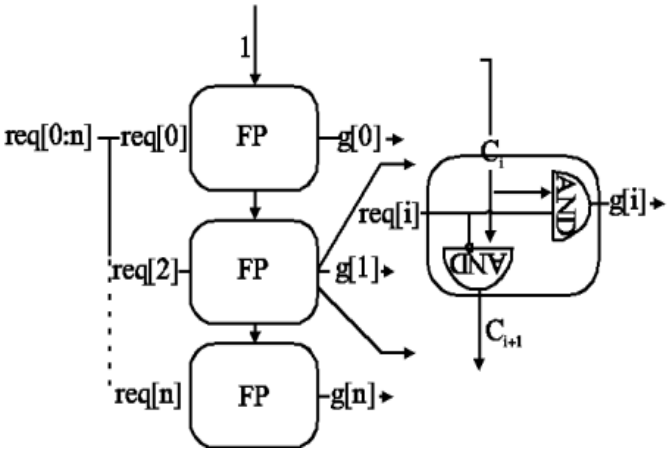

Fig. 2 Arbitration principle of FP arbiter

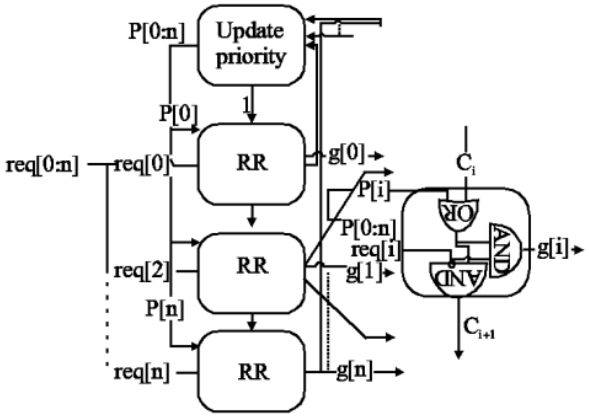

Fig. 3 Arbitration principle of RR arbiter

2.2. Dynamic Arbitration Strategies

2.2.1. Scheduling Theory and Optimization Theory

Scheduling theory and optimization theory provide foundations for arbitration strategies. Dynamic arbitration is a real-time scheduling problem. Decisions must be made dynamically. They depend on the system state and objectives. Common scheduling strategies include Earliest Deadline First (EDF), Shortest Processing Time First (SPTF), and Highest Value First (HVF). For example, Nesbit et al. used a network fair queuing algorithm. It employed the Earliest Virtual Completion Time (EVCT) priority strategy. This ensured service quality and fairness [14]. They defined the packet "slack"[15]. Slack is the number of cycles a packet can be delayed without affecting its execution time. It is a key metric. Prioritizing requests with low slack reduced application-level unfairness. It was 30.8% lower than the RR strategy.

2.2.2. Technical Pathways for Dynamic Arbitration

The core of dynamic priority adjustment lies in updating each master device's priority based on real-time bus status and device access characteristics, enabling on-demand scheduling. This ensures the priority hierarchy remains synchronized with the system state [16]. Typical application scenarios encompass bus environments with diverse devices, such as embedded systems and industrial automation controllers.

A hybrid arbitration strategy built upon this foundation integrates the strengths of multiple foundational approaches. It enables the construction of more suitable scheduling schemes for specific scenarios. Examples include the FP-RR dual-layer arbitration strategy and the AMBA multiprocessor bus arbitration scheme [17][18]. Compared to single-strategy solutions, this method demonstrates superior adaptability across diverse scenarios.

Machine learning and artificial intelligence are spreading. Novel intelligent arbitration strategies may use reinforcement learning and neural network prediction. They could analyze historical access patterns and states of the system to autonomously learn optimal scheduling policies. This approach promises powerful predictive scheduling capabilities, making it particularly well-suited for environments with significant load fluctuations.

3. Design and Implementation

This paper presents a four-factor weighted dynamic priority model using Verilog programming. In addition to fixed priority and wait time, the factors of “resource demand” and “urgent flag” have been added. A lower Resource_Need value is assigned higher priority in our model to encourage efficient use of bus resources by favoring devices with smaller transfer requests, thus optimizing overall resource allocation efficiency. The emergency flag is a binary indicator enabling rapid response to critical requests. It’s suitable for strict real-time scenarios such as industrial control and automotive electronics.

Figure 4 shows the overall architecture. It has four main components. Built on the non-preemptive priority queue system [19], a wait-time counter module was designed. This module has an independent counter for each device. It records the number of clock cycles waited since the last authorization. The parameter extraction module extracts relevant parameters for each device from the input configuration signal, including priority values, resource requirement values, and emergency flag status. The scoring calculation module is central. It computes each device's priority score using the four-factor weighted model.

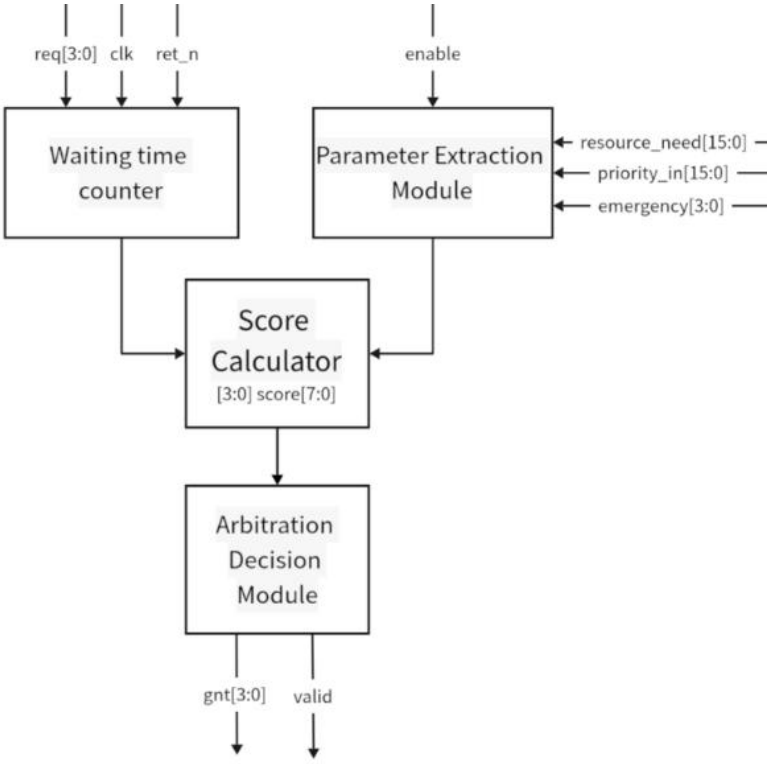

Fig. 4 Overall Architecture Diagram of the Arbitrator

To minimize "starvation," each device has a 4-bit counter. It increments only when a request is valid but unauthorized. It resets upon authorization or request invalidation. This mechanism quantifies unauthorized wait time. It is key to ensuring fairness.

The design uses two types of configurable parameters: system constant parameters and weight configuration parameters (Table 1). System constants adjust based on device count and response requirements. Weight parameters determine the influence of the four factors on the priority score. The system automatically calculates the total weight. Manual adjustment is not needed.

Table 1. System-configurable parameters

System Constant Parameters	Maximum wait time
	Maximum priority value
	Maximum resource demand
Weight Configuration Parameters	Weight_priority
	Weight_waittime
	Weight_resource
	Weight_emergency

The scoring calculation formula is as follows. The parameters and their information are shown in Table 2.

$$\text{Score} = (\text{Priority} \times W\_p + \text{Wait\_Time} \times W\_w + (\text{Max\_Resource} - \text{Resource\_Need}) \times W\_r + (\text{Emergency} \times W\_e) >> \text{Shift\_Bits}) \quad (1)$$

Table 2. Parameter information

Parameter	Conceptual Definition	Value Range	Weight Coefficient
Priority	Static priority of the device	0-15	W_P
Wait_Time	Number of cycles the device has been waiting	0-15	W_W
Resource_Need	Amounts of resources requested by the device	0-15	W_R
Emergency	Emergency flag	0-1	W_E

In the hardware implementation, the dynamic arbitrator employs the sum of weighting coefficients as the shift reference. This transforms division operations into right-shift operations, significantly reducing circuit complexity and latency. This approach conserves hardware resources while increasing the system's maximum operating frequency. Additionally, the design leverages the low-power characteristics of asynchronous hierarchical arbitration circuits [20]. Parameter extraction for devices can share the same logic unit, enabling resource reuse by sequentially processing different devices. At the final decision stage, authorization and output driving are split into two independent phases, allowing preceding and subsequent modules to operate in parallel, thereby boosting system throughput. These optimization techniques maximize hardware resource utilization while ensuring functional correctness.

4. Experimental Simulation and Result Analysis

4.1. Experimental Setup and Platform

The analysis involves evaluating their performance using FPGA-based methodologies. This paper employed Xilinx Zynq UltraScale+ MPSoC FPGA as the target platform. This device integrates a high-performance ARM processing system with programmable logic, making it suitable for prototyping complex digital systems. RTL code was written modularly. This ensures readability and maintainability. The development environment was Xilinx Vivado 2025.1. It handled the full design flow: synthesis, placement/routing, and timing analysis. Functional verification and timing simulation used a testbench on the ModelSim SE-64 10.7c platform.

4.2. Functional Verification Results

Functional verification is critical for ensuring design correctness. A comprehensive test platform using design constraints and testbench files was constructed. It covered normal operation, boundary conditions, and exceptional cases. The simulation waveform is shown in Figure 5.

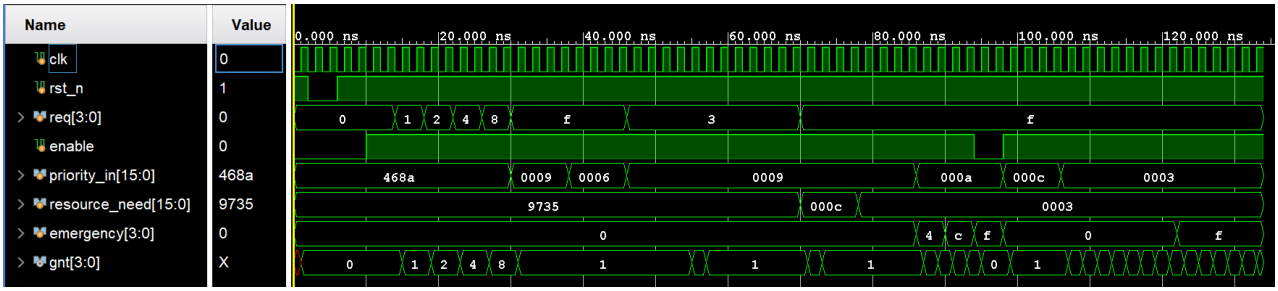

Fig. 5 Simulated waveform at a 500MHz clock frequency

Normal operation testing included single-device requests, simultaneous multi-device requests, and continuous requests. Results confirm that the arbitrator can accurately identify requests and make reasonable arbitration decisions based on the four-factor weighted model.

Boundary condition testing validated robustness under extreme scenarios. For instance, when all devices issued requests at the same time, the arbitrator fairly allocated bus resources according to the predefined algorithm.

Under extremely low load, it rapidly responded to individual requests, minimizing access latency. Tests under extreme weight configurations were also conducted. The design remained stable across various parameter combinations.

Anomaly testing simulated abnormal conditions like signal glitches and timing violations. By injecting faults, the design's fault tolerance and recovery mechanisms were validated. It demonstrated excellent robustness and the ability to maintain stable operation or recover rapidly from errors.

4.3. Performance Evaluation Results

Performance evaluation is essential for assessing quality of FPGA designs. This paper conducted a comprehensive evaluation of the dynamic arbitrator on the target board. For timing performance, static timing analysis tools were used to extract characteristics. These include setup and hold margins, maximum operating frequency, and authorization delay.

Under a 100 MHz clock and accounting for arbitration logic latency, the actual throughput is approximately 80-90 million transactions per second. Table 3 details the throughput and operational efficiency tested across different working modes.

Table 4 presents the power consumption analysis, which was conducted by assessing both static and dynamic power. Static power is primarily related to device technology and scale. Dynamic power is directly linked to circuit activity and operating frequency. The core static power consumption is 0.580 W, with I/O consumption at 0.083 W.

Table. 3 Throughput and Work Efficiency in Different Operating Modes

Working Mode	**Throughput (M transactions/s)**	**Efficiency**
Single Device Continuous Requests	89.3	100%
Four Devices Balanced Requests	89.3	100%
Emergency Preemption Mode	85.2	95.4%
Mixed Workload Scenario	82.1-87.6	92-98%

Table. 4 Hardware Resource Utilization

CLBLUTs	CLB Registers	CARRY8	CLB	LUT as Logic	Bonded IOB	BUFGCE
131	21	6	25	131	48	1

At a 500 MHz clock, average authorization latency stays below 1.2 ns. Logical resource consumption remains low. Its rapid response to urgent requests (1.03 ns) is particularly valuable. This validates its practical use in real-time systems.

Table.5 Performance Comparison with Three Static Arbitration Strategies

Performance Metric	Fixed Priority	Round-Robin Priority	Random Priority	Dynamic Priority
Average Grant Latency (ns)	0.74 ns	1.45 ns	1.62 ns	1.82 ns
Urgent Request Response Time (ns)	0.74 ns	1.45 ns	1.62 ns	1.34 ns
Static Power (W)	1.04	1.32	1.37	1.88
Total Power (W)	1.156	1.349	1.384	2.598
Resource Utilization	Very Low	Low	Low	High
Main Drawback	Prone to starvation	Cannot distinguish urgent requests	High randomness	High design complexity

Compared to several traditional strategies (Table 5), this design exhibits higher design complexity and power consumption. However, under high-load conditions where FP arbiters cause starvation or RR arbiters impact efficiency, this design maintains stable high system throughput through its dynamic scoring mechanism, demonstrating adaptability to non-standard operating environments.

The result validates the core value of this design—achieving real-time performance, fairness, and high throughput through intelligent resource scheduling while maintaining low power consumption

and compactness. It should be noted that the specific arbitration strategy selection should be determined based on the actual requirements of the application scenario.

5. Conclusion

This paper designs a multi-parameter weighted dynamic bus arbiter overhead through a hardware-optimized scoring algorithm and configurable weighting mechanism. It achieves a balance between high real-time performance, fairness, and low resource. FPGA experimental results demonstrate that the design can correctly grant arbitration rights across various workloads. However, current research still has limitations, including static weight configuration and insufficient scalability verification in ultra-large-scale SoCs. Future research will focus on introducing a reinforcement learning-based dynamic weight adjustment mechanism to enable online self-optimization of arbitration strategies. Concurrently, low-power optimization under advanced process technologies will be investigated to further enhance timing performance and energy efficiency.

References

[1] A. Mohanty and C. G N, "Comparative Analysis of Round Robin Arbiter and Weighted Round Robin Arbiter," 2024 Asia Pacific Conference on Innovation in Technology (APCIT), MYSORE, India, 2024, pp. 1-4.

[2] P. F. Orzechowski, C. Burger and M. Lauer, "Decision-Making for Automated Vehicles Using a Hierarchical Behavior-Based Arbitration Scheme," 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 2020, pp. 767-774.

[3] K. Khalil, B. Dey, A. Kumar and M. Bayoumi, "Adaptive Hardware Architecture for Neural-Network-on-Chip," 2022 IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS), Fukuoka, Japan, 2022, pp. 1-4.

[4] Jang, W., & Kim, D., "An Arbitration Control for an Ensemble of Diversified DQN Variants in Continual Reinforcement Learning," arXiv preprint arXiv:2509.04815v1, Sep. 2025.

[5] Zhu, Y. Q. and X. J. Hou, "Design of DMA data stream arbiter based on bit-map algorithm," Journal of Henan University of Technology (Social Science Edition), vol. 41, no. 04, pp. 116-122, 2025.

[6] Liu, H. L., Q. H. Yan and J. L. He, "Arbiter PUF bias control method based on response grouping," Modern Electronics Technique, vol. 47, no. 09, pp. 104-108, 2024, doi:10.16652/j.issn.1004-373x.2024.09.019.

[7] Li, X., "Design and implementation of arbiter PUF based on FPGA," Master's thesis, Harbin University of Science and Technology, Harbin, China, 2024.

[8] Zhou, G. H., D. C. Zou and X. C. Lu, "Design and implementation of multi-priority universal route arbitrator," Journal of Chinese Computer Systems, vol. 41, no. 03, pp. 593-597, 2020.

[9] Xu, Z. S., "The design to dynamic weight arbitrator based on network-on-chip," Journal of Changchun Institute of Technology (Natural Science Edition), vol. 20, no. 04, pp. 55-58, 2019.

[10] Legkikh, A. M., "Comparison of Round Robin and Wrapped Wavefront Arbitration Algorithms Characteristics on the Example of Crossbar Switch Arbitration," 2022 Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), Saint Petersburg, Russia, 2022, pp. 162-165.

[11] Ganiee, S. A., Ganiee, S. A., & Dar, J. R., "FPGA Implementation of an 8-bit 4×4 Crossbar Switch for Multiprocessor System-on-Chip (SoC) Using Round Robin Arbitration Algorithm," International Conference on VLSI and Embedded Systems (ICVES), Srinagar, India, 2023, pp. 116-122.

[12] Tiwari, R. Chandraker and N. Goel, "Comparative analysis of different lottery bus arbitration techniques for SoC communication," 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), New Delhi, India, 2016, pp. 495-499.

[13] D. Kutuzov, A. Osovsky, D. Starov, O. Stukach, N. Maltseva and D. Surkov, "Crossbar Switch Arbitration with Traffic Control for NoC," 2022 International Siberian Conference on Control and Communications (SIBCON), Tomsk, Russian Federation, 2022, pp. 1-5.

[14] Nesbit, K. J., N. Aggarwal, J. Laudon and J. E. Smith, "Fair Queuing Memory Systems," The 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06), 2006.

[15] Das, R., O. Mutlu, T. Moscibroda and C. R. Das, "Aérgia: Exploiting Packet Latency Slack in On-Chip Networks," 2010 International Symposium on Computer Architecture (ISCA'10), Saint-Malo, France, 2010, pp. 106-116.

[16] Nedbailo, Y. A., "Inverse Weight Based Arbitration Algorithm for On-Chip Networks," 2024 8th International Conference on Information, Control, and Communication Technologies (ICCT), Moscow, Russia.

[17] L iu, L., X. F. Zhou, Z. M. Zhu, D. Zhou and Y. T. Yang, "Novel bus arbiter with the two-level arbitration mechanism," Journal of Xidian University (Natural Science Edition), vol. 44, no. 01, pp. 12-17, 2017, doi:10.3969/j.issn.1001-2400.2017.01.003.

[18] V. M. I. R. Rokon, S. M. A. Motakabber, A. H. M. Zahirul Alam, M. Hadi Habaebi and M. A. Matin, "Smart Arbitration System for Multiprocessor AMBA Interface in System on Chip," 2021 8th International Conference on Computer and Communication Engineering (ICCCE), Kuala Lumpur, Malaysia, 2021, pp. 208-213.

[19] S. K. Mandal, R. Ayoub, M. Kishinevsky, M. M. Islam and U. Y. Ogras, "Analytical Performance Modeling of NoCs under Priority Arbitration and Bursty Traffic," in IEEE Embedded Systems Letters, vol. 13, no. 3, pp. 98-101, Sept. 2021.

[20] J. Guo，G. Lin，B. Wang，Z. Yu and S. Xiao, “Asynchronous Arbitration Circuit Optimization for Multicore Neuromorphic Processors”, 2024 IEEE 17th International Conference on Solid-State &; Integrated Circuit Technology (ICSICT), Zhu Hai, China, 2024, pp. 1-3