The Design of Fixed Priority and Round Robin Arbitrator

Wenbo Wang *

Department of Materials Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

* Corresponding Author Email: wangwenbo4253@gmail.com

Abstract. System performance, especially in multiprocessor systems, largely depends on the efficiency of the bus architecture. In a system-on-a-Chip (SoC), the arbiter is a key technology in the shared bus structure, used to solve the conflict problem when multiple controller devices simultaneously request bus resources. The fixed priority arbiter has the advantages of low resource consumption and is suitable for scenarios with strict real-time requirements. The round robin arbitrator has the characteristic of high arbitration fairness. This paper designs two different designs of arbitrators, including two fixed priority arbitrators implemented through different algorithms and one round robin arbitrator. Fixed priority arbitrators can ensure that critical tasks receive responses within the same clock cycle when signals arrive, which is crucial for scenarios such as interrupt handling and high-speed data transmission. Efficient arbitration algorithms, such as a round robin arbitrator, can balance the access opportunities of various devices, avoid low-priority devices waiting for a long time, and thus improve the overall throughput of the system. In this study, the fixed priority arbitrator optimizes the algorithm to achieve the output of the arbitration result when the clock signal arrives. The round robin arbitrator implements a fairer arbitration mode based on the fixed priority arbitrator. Ensure that every signal has the opportunity to be arbitrated.

Keywords: SoC; fixed priorty; round robin arbitrator.

1. Introduction

In contemporary highly integrated digital system-on-a-chip (SoC) designs, computational cores, memory controllers, and various peripheral modules interact through shared communication infrastructures (e.g., buses, network-on-chip interconnects) to exchange data and instructions. This shared-resource architecture demonstrates significant advantages in enhancing system efficiency, reducing power consumption, and lowering costs [1,2].

System-on-a-Chip (SoC) is a highly integrated solution that consolidates computing cores, storage systems, communication interfaces, and peripheral devices onto a single chip. Its core architecture typically comprises the following components: Computing Core: Includes CPUs, GPUs, NPUs, etc., where ARM architecture dominates mobile and embedded markets due to its superior energy efficiency ratio [3, 4]. Storage Subsystem: Composed of PFlash (non-volatile) and SRAM (volatile), supporting external Flash or DRAM expansion to enhance capacity. Communication buses such as AXI, AHB, and APB, as well as Network-on-Chip (NoC), enable high-speed data transfer between components. Peripheral I/O includes high-speed interfaces like DMA and PCIe. and low-speed interfaces.

In the Advanced High-performance Bus (AHB) architecture, the arbiter serves as a critical component of the bus matrix, with its primary functions including: Controller Device Priority Management and Address/Data Bus Control, which determine which controller device (such as the CPU, DMA, peripheral controller) gains bus access rights through either fixed-priority or round-robin arbitration strategies. For instance, the AHB bus matrix in the STM32F407VET6 microcontroller employs a fixed-priority scheme, where the DMA controller is granted higher priority than CPU background tasks [5].

This architecture also introduces a fundamental challenge: when multiple controller devices simultaneously request access to the same agent device or shared communication channel, how to efficiently and orderly allocate access permissions to avoid data conflicts, ensure system functional correctness, and optimize overall performance [6]. The key component addressing this challenge is

the arbiter. The quality of design directly determines the utilization efficiency of system resources and the service quality of requests from all controller devices [7].

Fixed priority arbitrators can ensure that critical tasks receive responses within the same clock cycle when signals arrive, which is crucial for scenarios such as interrupt handling and high-speed data transmission. Research has shown that optimized arbitrators can reduce latency to the nanosecond level. Efficient arbitration algorithms, such as a round robin arbitrator, can balance the access opportunities of various devices, avoid low-priority devices waiting for a long time, and thus improve the overall throughput of the system. Experimental data show that round robin arbitrators can increase bus utilization by 15-20%. The combination of arbiter and Dynamic Voltage Frequency Adjustment (DVFS) technology can dynamically adjust the working status of modules according to task priorities, achieving energy efficiency optimization. In some SoC designs, the arbiter contributes approximately 8% of power savings. In summary, the importance of arbitrators in SoC systems lies in their role as the core component of resource management, ensuring orderly access to shared buses by multiple controller devices and balancing real-time, fairness, and system efficiency through different arbitration strategies. Directly affecting the performance, power consumption level, and reliability of SoC [8].

The organization of this paper is as follows: Section 2 provides a detailed analysis of two distinct fixed-priority arbiter algorithms and one round-robin priority arbiter algorithm [9]. Section 3 presents experimental and simulation results. Section 4 concludes the paper with a summary of the entire work.

2. Algorithm Implementation of Arbitrator

2.1. fixed priorty arbitrator

When designing a fixed-priority arbiter, the first step is to clearly define the design requirements, including a Configurable number of request inputs for the arbitration module. In this paper, there are four requests requiring arbitration, namely Request 0, Request 1, Request 2, and Request 3. The core challenge in fixed-priority arbiter design lies in identifying the highest-priority request among multiple competing inputs. For example, when req = 4'b1011, the corresponding grant output would be 4'b0001. When implementing the first fixed-priority arbiter design, the arbiter is configured such that its priority hierarchy is always Request 0 > Request 1 > Request 2 > Request 3 under all conditions. In FPGA implementations using Verilog, this can be readily achieved by sequentially checking these four data bits from the lowest bit to the highest bit for non-zero values. The first non-zero bit encountered determines the output group. The above algorithm is shown in Table 1.

Table 1. Algorithm 1:fixed priority arbitrator.

Inputs:
CLK:(clock signal)
RSTN:active-low reset signal
REQ [3:0]:4-bit request signals, where REQ [0] has the highest priority
Output:
GRANT [1:0]:2-bit grant signal, indicating the index of the currently granted request
1. On clock rising edge:
2. **If** RSTN is low:
3. Force GRANT = 0 (reset state)
4. **Else** check REQ signals in priorty
5. **If** REQ [0] is active → GRANT = 0
6. **If** REQ [1] is active → GRANT = 0
7. **If** REQ [2] is active → GRANT = 0
8. **If** REQ [3] is active → GRANT = 0
9. **If** no requests → GRANT = 0

In Table 2, the original data value is first decremented by one, followed by a bitwise inversion of the decremented result. Finally, the inverted data undergoes a bitwise AND operation with the original data. The position of the resulting '1' determines the arbiter's output. For example, with an input data of '1110':

Table 2. Algorithm 2:Improved fixed priority arbitrator.

Inputs:
CLK:(clock signal)
RSTN:active-low reset signal
REQ [3:0]:4-bit request signals, where REQ [0] has the highest priority
Output:
GRANT [1:0]:2-bit grant signal, indicating the index of the currently granted request
1. On clock rising edge:
2. **If** RSTN is low:
3. Force GRANT = 0 (reset state)
4. **Else** TEMP ← REQ - 1
5. MASK ← BITWISE_NOT(TEMP)
6. GRANT ← REQ BITWISE_AND MASK

Step1: Decrement the original value by one to obtain '1101', Step2:Perform bitwise inversion on '1101' to yield '0010', Step3:Execute a bitwise AND between the inverted data (0010) and the original data (1110), resulting in '0010'The arbitration result is the first '1' bit encountered from the least significant to the most significant bit (Request 0 in this case). The entire process is shown in Figure1.

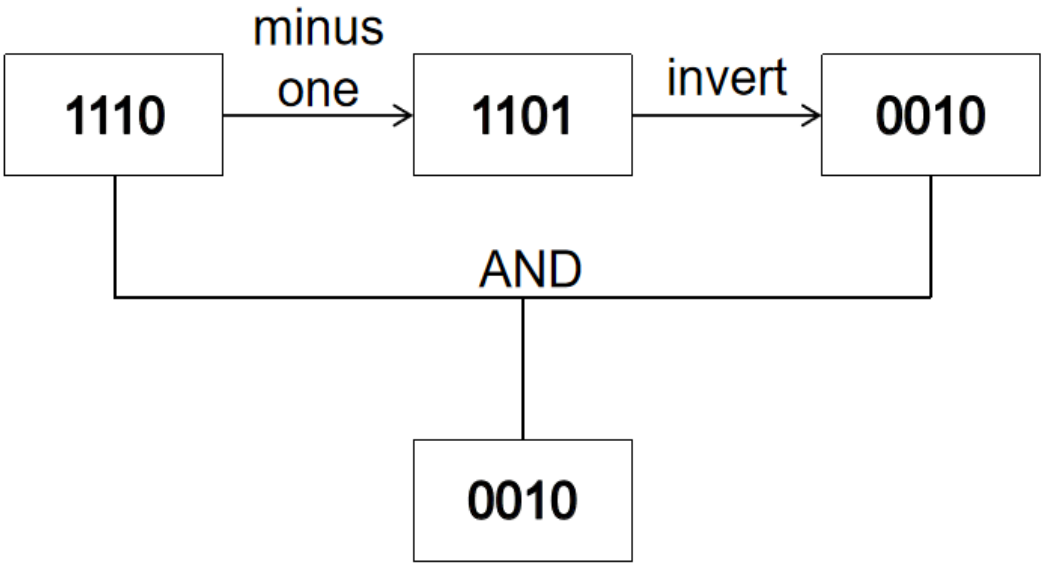

Fig. 1 Example of Algorithm Implementation.

2.2. round robin arbitrator

The algorithm of the round robin arbitrator is shown in Table 3. To prevent signal starvation where specific signals may monopolize the arbiter while others remain unresponsive, this paper proposes an enhanced round-robin priority arbiter algorithm. In this approach, all signals are granted equal priority, with each signal sequentially obtaining authorization. For instance, when four requests are simultaneously submitted to the arbiter, Request 0 initially holds the highest priority. After servicing Request 0, the arbiter elevates Request 1 to the highest priority, and so forth. This cyclical prioritization ensures every signal has an opportunity to achieve the highest priority, thereby improving arbitration fairness [10].

Table 3. Algorithm 3:The round robin arbitrator.

Inputs:
CLK:(clock signal)
RSTN:active-low reset signal
REQ [3:0]:4-bit request signals, where REQ [0] has the highest priority
Output:
GRANT [1:0]:2-bit grant signal, indicating the index of the currently granted request

```
On clock rising edge:
If RSTN is low:
    Force GRANT =d3 (reset state)
Else NEXT ← (GRANT + 1) MOD 4 // Cyclic increment
        for i IN 0 TO 3:
            CURRENT ← (NEXT + i) MOD 4
            If REQ[CURRENT]:
                GRANT ← CURRENT
                BREAK
END MODULE
```

3. Results and Analysis

3.1. Parameter Setting for Results

To verify the correctness of the arbiter design, this paper employs Modelsim software to simulate the implemented arbiter algorithm. Four test datasets are used for validation. According to the correct logic, the two arbiters should produce the arbitration results as shown in the following table. The arbitration results output by the fixed priority arbitrator are shown in Table 4, and the arbitration results output by the round robin arbitrator is shown in Table 5. If the simulation results match the expected outcomes, the design of the arbiter is considered successful.

Table 4. Expected result (fixed priorty).

Parameter	Results (fixed priorty)
1001	Request 0
1101	Request 0
1110	Request 1
1010	Request 3

Table 5. Expected result (round robin).

Parameter	Results (round robin)
1101	Request 0
1111	Request 1
1110	Request 2
1100	Request 3

3.2. Simulation results of fixed priority arbitrator

The simulation results clearly indicate that when the input values are assigned to '1001' or '1101', the arbiter prioritizes responding to the signal request from the first module and outputs its corresponding data Request 0. For input values' 1110' or '1010', the arbiter gives priority to the second module's request and consequently outputs its data Request 1. When the input is '1000', the arbiter preferentially serves the fourth module's request, outputting Request 3. These results demonstrate perfect agreement between the simulation outcomes and the expected behavior, thereby implementing a fixed-priority digital arbiter with the established priority sequence: Request 0 > Request 1 > Request 2 > Request 3. The experimental results are shown in Figure 2.

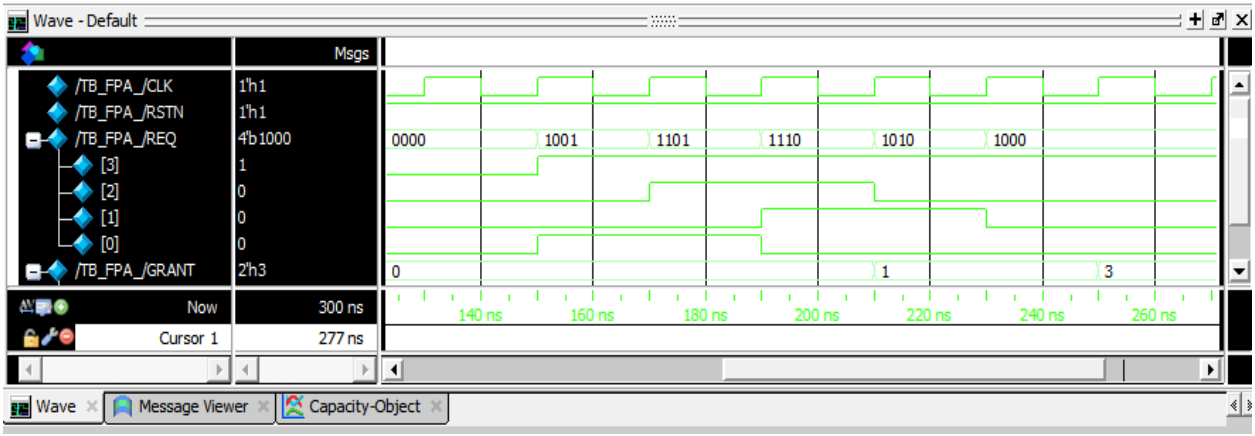

Fig. 2 Simulation result of fixed priority arbitrator.

The simulation results of the improved fixed-priority arbiter algorithm are shown in Figure 3. According to the simulation results, the improved algorithm correctly executes the arbitration order with fixed priorities of Request 0 > Request 1 > Request 2 > Request 3. Compared to the previous fixed-priority arbiter algorithm, the improved algorithm outputs the arbitration result within the same clock cycle. In contrast, the previous version required waiting for one clock cycle to obtain the arbitration result at the rising edge of the next clock cycle. The improved algorithm outperforms the previous one in terms of computational speed and resource utilization.

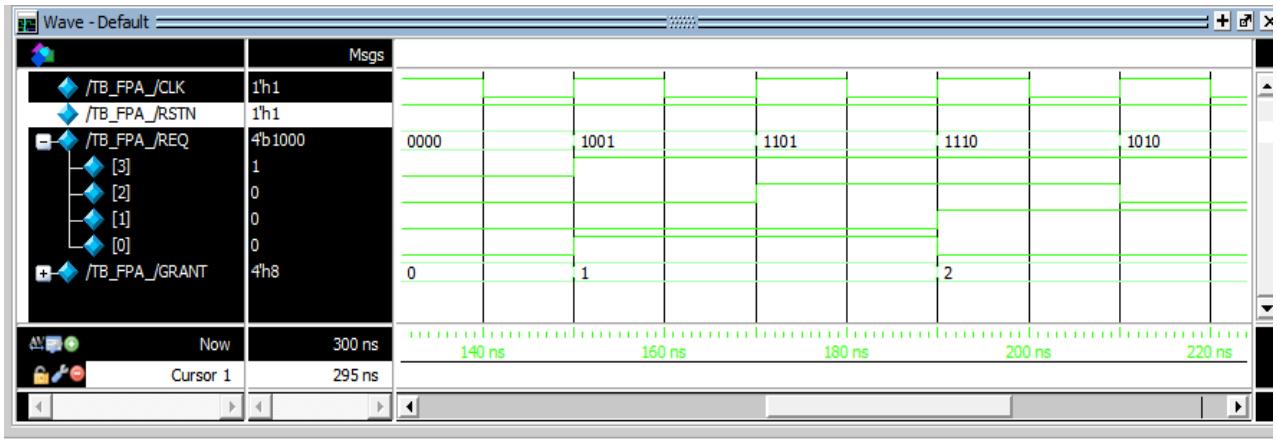

Fig. 3 Simulation result of the improved fixed-priority arbitrator.

3.3. Simulation results of the round robin arbitrator

The simulation results of the round robin arbitrator algorithm are presented in Figure 4. As demonstrated by the simulation results, unlike the fixed priority arbitrator, when the input value is '1101', the arbiter prioritizes responding to the first bit's signal request. It outputs the data of the first module, Request 0. When the input value is '1111', despite the first bit remaining 1, the arbiter responds to the second bit's arbitration request and subsequently outputs the second bit's data, Request 1. For the input value '1110', the arbiter prioritizes the third bit's signal request and outputs Request 2. Finally, when the input value is '1100', the fourth bit holds the highest priority, resulting in the output of Request 3.

The simulation results align perfectly with the expected outcomes. This algorithm improves upon the fixed priority arbitrator by ensuring each signal has an opportunity to be arbitrated, thereby enhancing the fairness of arbitration results. However, the simulation also reveals that the arbitration result is delayed by one clock cycle. Future research should focus on further optimizing the algorithm to enable output within the same clock cycle as the signal arrival.

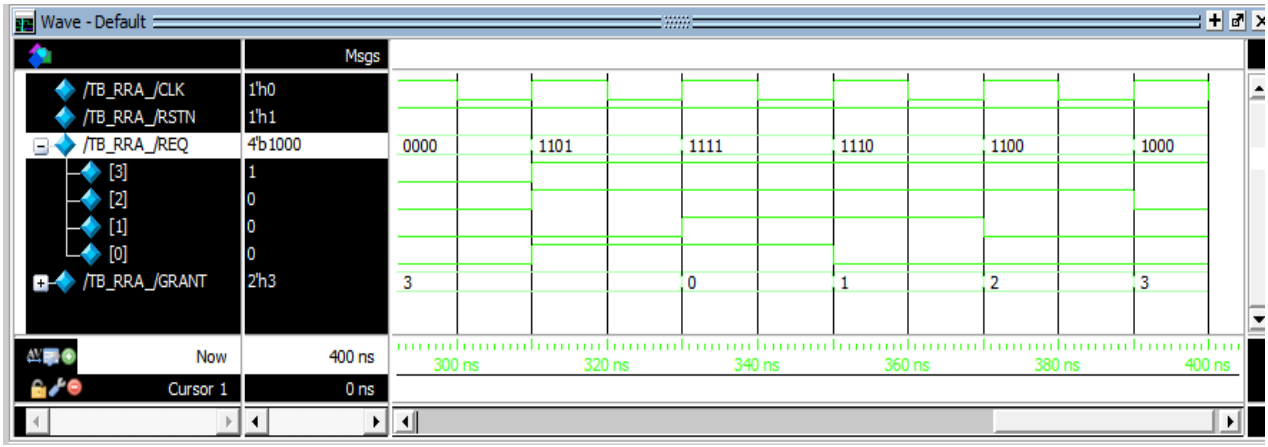

Fig. 4 Simulation result of the round robin arbitrator.

4. Conclusion

This paper designs a fixed priority arbitrator implementation with the priority order: Request 0 > Request 1 > Request 2 > Request 3. When multiple requests arrive simultaneously, the arbiter always grants access according to the predefined priority sequence, ensuring correct arbitration results. Through algorithmic optimization, the arbitration output can be generated within the same clock cycle as the signal arrival. However, the fixed priority arbitrator's primary drawback lies in the lack of fairness. To address this, this paper proposes a round robin arbitrator based on the fixed priority design. This arbitrator implements priority rotation by assigning the arbitration priority to the next module in sequence after each arbitration result, ensuring all signals have equal opportunity to access the resource and thereby improving fairness.

References

[1] Huang, Y. J., Chen, Y. H., Yang, C. K., & Lin, S. J. Design and implementation of a reconfigurable arbiter. In Proceedings of the 7th WSEAS International Conference on Signal, Speech and Image Processing, September, pp. 100-105.

[2] Li, H., Zhang, M., Zheng, W., & Li, D. An adaptive arbitration algorithm for SoC bus. In the 2007 International Conference on Networking, Architecture, and Storage, 2007, pp. 245-246.

[3] Doifode, N., Padole, D., & Bajaj, P. R. Design and performance analysis of efficient bus arbitration schemes for on-chip shared bus multiprocessor soc. Journal of Computer Science and Network Security, 2008, pp. 250-255.

[4] Akesson, B., Steffens, L., & Goossens, K. Efficient service allocation in hardware using credit-controlled static-priority arbitration. In 2009, 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2009, pp. 59-68.

[5] Pyoun, C. H., Lin, C. H., Kim, H. S., & Chong, J. W. The efficient bus arbitration scheme in the SoC environment. In The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications, 2003, pp. 311-315.

[6] Akesson, B., Steffens, L., Strooisma, E., & Goossens, K. Real-time scheduling using credit-controlled static-priority arbitration. In 2008, 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2008, pp. 3-14.

[7] Rigaud, J. B., Quartana, J., Fesquet, L., & Renaudin, M. Modeling and design of asynchronous priority arbiters for on-chip communication systems. In SOC Design Methodologies: IFIP TC10/WG10.5 Eleventh International Conference on Very Large Scale Integration of Systems-on-Chip (VLSI-SOC'01), Montpellier, France, December 3–5, 2001, pp, 313-324.

[8] Zitouni, A., & Tourki, R. Arbiter synthesis approach for SoC multi-processor systems. Computers & Electrical Engineering, 34(1), 2008, pp, 63-77.

[9] Srinivasan, P., Ahmadinia, A., Erdogan, A. T., & Arslan, T. Power Evaluation of the Arbitration policy for different On-Chip Bus-based SoC platform. In 2007 IEEE International SOC Conference, September 2007, pp. 159-162.

[10] Zhikui, L., & Zhenger, Y. An optimized bus arbitration scheme in multiprocessor SoC. In China-Japan Joint Microwave Conference, April 2011, pp. 1-4.