Research Article

High Priority Arbitration for Less Burst Data Transactions for Improved Average Waiting Time of Multi-Processor Cores

Ahmed Noami*, Boya Pradeep Kumar and Chandra Sekhar Paidimarry
Department of Electronics and Communication Engineering, College of Engineering, Osmania University, Hyderabad, India

Abdullah Alahdal
Department of Computer Science and Engineering, College of Engineering, Osmania University, Hyderabad, India

Nada Safi
Department of Computer Science, College of Science, Osmania University, Hyderabad, India

* Corresponding author. E-mail: mrahmedyahya883@gmail.com DOI: 10.14416/j.asep.2021.06.001
Received: 7 November 2020; Revised: 4 March 2021; Accepted: 23 April 2021; Published online: 4 June 2021

Abstract
The multi-processor cores in SoC which have high burst data transactions can play a critical role while accessing the shared resources such as the off-chip memory. These processor cores can starve other processor cores that have less burst data transactions while accessing the same shared resources. The starving issue of other processor cores leads to degrade the entire system performance of the SoC. However, the arbiter architecture in the SoC design plays the best solution to manage different processor core requests and granting one of them to access the shared resources according to different scheduling algorithms. In this paper, we have designed AXI interconnect, which includes arbiter architecture to connect four processor cores represented by the AXI masters and the off-chip memory represented by the salve. Each processor core (AXI Master) uses the AXI4 interface protocol to improve the system performance and the arbiter based on the static fixed-priority algorithm to improve the average waiting time for all the processor cores. The SoC design architecture is modeled in System Verilog HDL; simulation and synthesis are done by using the Vivado tool and FPGA ZYNQ-7 ZC702 Evaluation Board (xc7z020clg484-1).

Keywords: SoC, AMBA AXI Interface, AXI Interconnect, Arbiter

1 Introduction

Nowadays the modern system on chip (SoC) includes several processor cores on the same die. These processor cores are widely used for different purposes such as embedded, networks, and digital signal processing [1]. There are so many issues while these processor cores communicate together or while accessing the shared resources such as on-chip memory, off-chip memory, and peripherals. The communication latency is the main issue while these processor cores using improper interface protocol to communicate together or using unfair memory management to access the shared off-chip memory [2]–[4]. The ARM Advanced Microcontroller Bus Architecture (AMBA) introduced different types of interfaces. The first AMBA interface version was

Please cite this article as: A. Noami, B. P. Kumar, C. S. Paidimarry, A. Alahdal, and N. Safi, "High priority arbitration for less burst data transactions for improved average waiting time of multi-processor cores," *Applied Science and Engineering Progress*, vol. 14, no. 3, pp. 397–405, Jul.–Sep. 2021, doi: 10.14416/j.asep.2021.06.001.

the advanced peripheral bus and the advanced system bus. The second version of the AMBA2 interface was AMBA high-performance bus (AHB) that has only two independent channels for write and read transactions, does not support burst mode capability (each data transaction needs an independent address). The third version of the AMBA3 interface was an Advanced eXtensible Interface (AXI). AMBA AXI3 interface protocol is the high-performance communication interface and it supports the burst length of data transactions up to 16 beats for a single address [5]. The fourth version of the AMBA4 interface includes different kinds of interface: AXI4-Lite interface, AXI4 full map interface, and AXI4-Stream interface. The AXI4-Lite interface just supports the single burst mode with 32-bits data size, while the AXI4 full map interface supports different bursts mode up to 256 beats with different data sizes like 32-bits or 64-bits, and it has five independent transaction channels: write address channel, write data channel, write response channel, read address channel, and read data channel [6], [7]. The AXI4-Stream is designed only to transfer the unidirectional streams of data such as pixel stream for image/ video. Beside selecting the proper interface communication to improve the SoC performance, the memory management of multi-processor cores is a very important part of the entire SoC when the processor cores access the off-chip memory. When the multi-processor cores need to access the shared resource (off-chip memory) at the same time, each processor core may have a different data size and burst length. The processor core that has high burst data transaction can starve the other processor cores that have less burst data transaction if it gets the access first. The best memory controller appears in this issue and how it can prevent this starvation as possible. The main component of the memory controller that can manage all the requests from different processor cores with different sizes and burst lengths is the arbiter. It is an architecture that decides who the processor core can access the off-chip memory first, and the decision depends on the different memory scheduling algorithms [8].

2 Literature Survey

The proper SoC architecture and communication interface on the same die can lead to high-performance SoC and get fairness in terms of average waiting time while accessing the shared resources among all the processor cores. Many previous studies designed multi-processor cores architecture using different communication interfaces and arbiters with different scheduling algorithms. Shrivastava *et al.* [9] designed the arbiter architecture to manage the requests for multi-processor cores to access the shared resources. However, this study did not consider the data burst length for each processor core and the data for each processor core was represented by a single clock cycle. Khanam *et al.* [10] the SoC arbiter is designed to manage all requests from the different processor cores to access the shared resources. However, this study used less-performance communication interface protocol: Advanced microcontroller bus architecture advanced system bus (AMBA ASB). According to Ingle *et al.* [11] and Bhat *et al.* [12], the SoC arbiter is designed to manage the requests for many processor cores to access the shared resources. These designs used a high-performance communication interface protocol: Advanced High-performance Bus (AMBA AHB). However, there are many drawbacks in this type of interface such as limited throughput compared to the AMBA AXI interface protocol. Tiwari *et al.* [13] the SoC arbiter for multi-processor cores is considered using high-performance protocol (AMBA AXI4-Lite). However, this communication interface supports only a single burst mode of 32-bits data transaction, and each processor core takes equal time to complete the transaction which means the fairness among all the processor cores is very high but the performance is less. Furthermore, Noami *et al.* [14] proposed a united multi-core memory controller for four-processor cores to write/ read transactions using the AXI4-Lite interface without considering the arbiter architecture. This means that each processor core can initiate write/ read in 32-bits data size with single burst capability and all processor cores takes the same number of clock cycles to complete the transactions. Noami *et al.* [15] also proposed the AXI4 interface protocol for multi-core memory controller to write/ read transactions in different burst lengths without considering the arbiter architecture. The different burst lengths were 0f, 07,03, and 01 for the first, second, third, and fourth processor core respectively. Without considering the arbiter architecture, the default arbitration is done, which means that the first processor core should access first.

In this scenario, the write/read transactions will take more time to complete which led to an increase in the average waiting time for all the processor cores.

In this work, the AXI interconnect with considering the arbiter based on the static fixed-priority arbitration is proposed. The AXI interconnect architecture is designed to connect four processor cores and off-chip memory. All the four-processor cores use the AXI4 interface protocol with different burst lengths to write/ read to/ from the off-chip memory. Each processor core can initiate the transaction in different burst lengths by using the high-performance AXI4 interface protocol. However, only the processor core that has less burst data transaction such as 01 burst length gets a high priority to access the off-chip memory to improve the average waiting time for all the four processor cores which leads to improve the entire SoC performance as compared with the existing work.

3 Proposed Model

The AXI interconnect architecture which includes the arbiter based on the static fixed-priority algorithm is proposed as shown in Figure 1. The AXI interconnect connects four processor cores which are represented by AXI masters and off-chip memory which is represented by the slave. All the four-processor cores can initiate the write/ read transactions by using the AXI4 interface protocol in 32-bits data size with different burst lengths capability. All the four-processor cores can initiate the write/ transactions to/ from the off-chip memory at the same time. Each processor core can write and read in different burst lengths such as 0f, 07, 03, and 01 (i.e. 16, 8, 4, and 2 beats of data transactions) for the first, second, third, and fourth processor cores respectively. The AXI interconnect will receive these four different burst length requests from the four-processor cores and the arbiter architecture will decide among all the requests which processor core can use the shared interface to access the off-chip memory first according to a static fixed-priority algorithm. At the program time, we used a static fixed-priority algorithm to assign the priority for all the processor cores (Masters). The processor core that has less burst data transactions such as 01 (i.e. 2 beats of data transactions) will get the high priority to accessing the off-chip memory and the processor core that has high burst data transactions such as 0f (i.e. 16 beats of data transactions) will get the low priority to accessing the off-chip memory. The main goal of these priorities is to improve the average waiting time for all the processor cores which leads to improving the entire SoC performance.

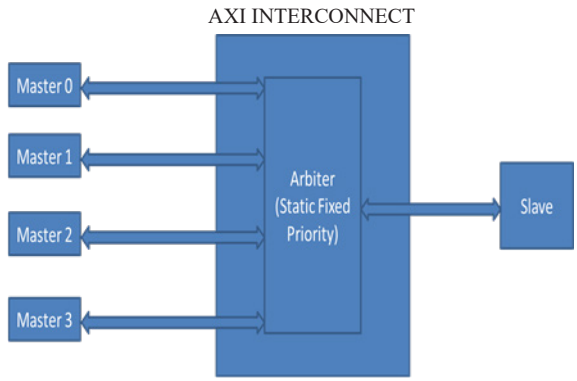

Figure 1: Proposed model.

3.1 *Finite state machine of AXI arbiter*

The finite state machine (FSM) of the AXI Arbiter based on the static fixed-priority algorithm (SFP) is discussed in this section and shown in Figure 2. In the beginning, the idle state is there when the AXI arbiter not receiving any request from the AXI Masters (processor core) to decide which AXI master can use the shared interface. Thus, accesses first and starts the transaction. At the time of clock start and reset signal raising high, any AXI master can send a request to use the shared interface and start the transaction. If any two processor cores or more send the requests at the same time to use the shared interface and start their transactions, the arbiter based on static fixed-priority algorithm grants the processor core that has the high priority scheduling (at the program time, the high priority is given to the processor core that has less burst data transaction). After granting the shared interface to the selected processor core, the processor core can start the transaction. Then, at the time transaction is completed of the first granted processor core, the arbiter grants the shared interface to the second processor core that has less burst data transactions. The arbiter will continue granting the shared interface based on the static fixed-priority algorithm for the remaining processor cores till they are completed. When the last processor core finished the transactions, the state will return to the idle state as shown in Figure 2 below.

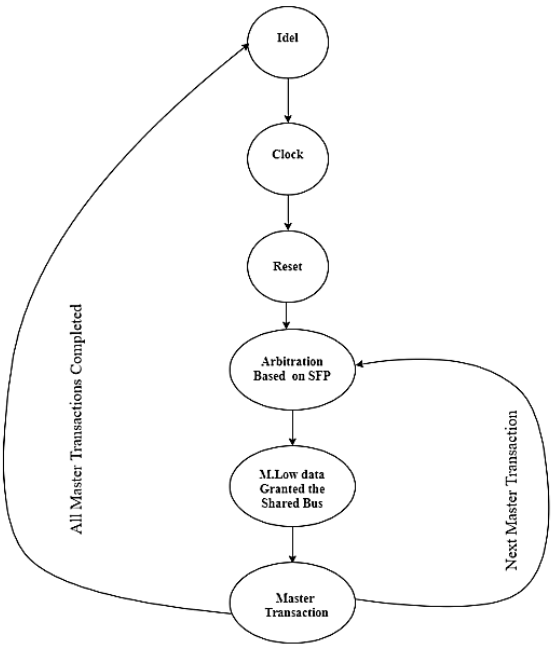

Figure 2: FSM of the AXI arbiter based on SFP.

In this work, we proposed this model and considered the arbiter architecture to improve the average waiting time of the existing work. The less burst data burst length transactions need a less number of cycles to complete the transactions comparing with high burst data burst length transactions, that is why the processor core that has less burst data transactions will get high priority to accessing the off-chip memory first than the other processor cores in this work.

4 Simulation Results

The simulation results of the proposed model are discussed in this section. The AXI interconnect with considering the static fixed-priority scheduling algorithm is implemented to connect four-processor cores which are represented by four AXI Masters and off-chip memory which is represented by the Slave as shown in Figure 1 of the proposed model. In this work, we focus on the main component of the AXI interconnect which is called the arbiter based on the static fixed-priority scheduling algorithm. The arbiter is a digital logic architecture that receives all the requests from multi-processor cores and gives the final decision based on the static fixed-priority scheduling algorithm, the processor core that has less burst data length (high priority) will access the off-chip memory first. The four processor cores initiate the transactions using the AXI4 interface protocol in 32-bits data size with different burst lengths capability.

The proposed design of this work is considers the arbiter architecture based on the static fixed-priority scheduling algorithm to improve the average waiting time for all the processor cores and also using the AXI4 interface protocol to improve the entire SoC performance. The existing work [14], designed a united multi-core memory controller for four-processor cores using the AXI4-Lite interface protocol without considering the arbiter architecture. That is why the first processor core always accesses first (i.e. default arbiter which has the same priority for all the processor cores). Also, while using the AXI4-Lite interface protocol each processor core has a single burst length and the same clock cycles to complete the transaction. The more waiting times for all the processor cores happen when the SoC does not consider the arbiter architecture while the multi-processor cores SoC design using the AXI4 interface protocol with different burst lengths such as 0f, 07, 03, and 01 (i.e. 16, 8, 4, and 2 beats of data transactions) for the first, second, third, and fourth processor cores respectively [15] as shown in the simulation results of Figure 3 and 4.

The AXI4 full memory map interface protocol has five standard channels: three channels for write transactions and two channels for read transactions. In this simulation results section, we implemented only the write transaction channels for all the processor cores while accessing the off-chip memory.

From the simulation results, we observed that the Figure 3 shows the write transaction channel signals of the first and second processor cores in 32-bits data size with different burst lengths of 0f and 07 respectively. Figure 4 shows the write transaction channel signals of the third and fourth processor cores in 32-bits data size with different burst lengths of 03 and 01 respectively [15]. In Figure 3, the first processor core started the write transaction signals at the time 30 ns when the clock and reset signals are high. At the time 410 ns, the m0_BVALID (i.e. Slave generates this signal when the write response on the bus is valid) and m0_BREADY (Master generates this signal when it can accept a write response) signals are high which indicates that the first processor core

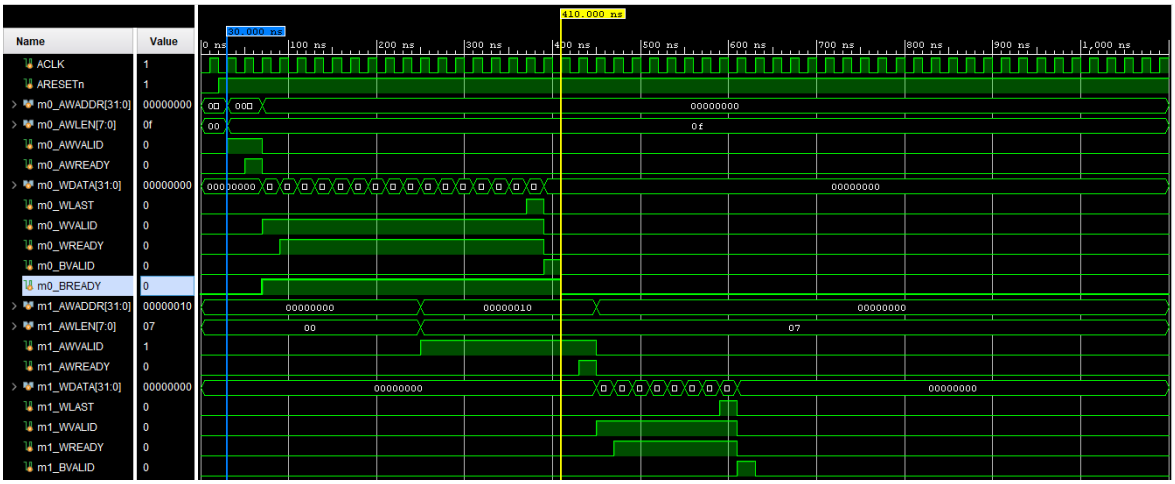

Figure 3: Write channel signals of the first and second processor cores without considering the arbiter.

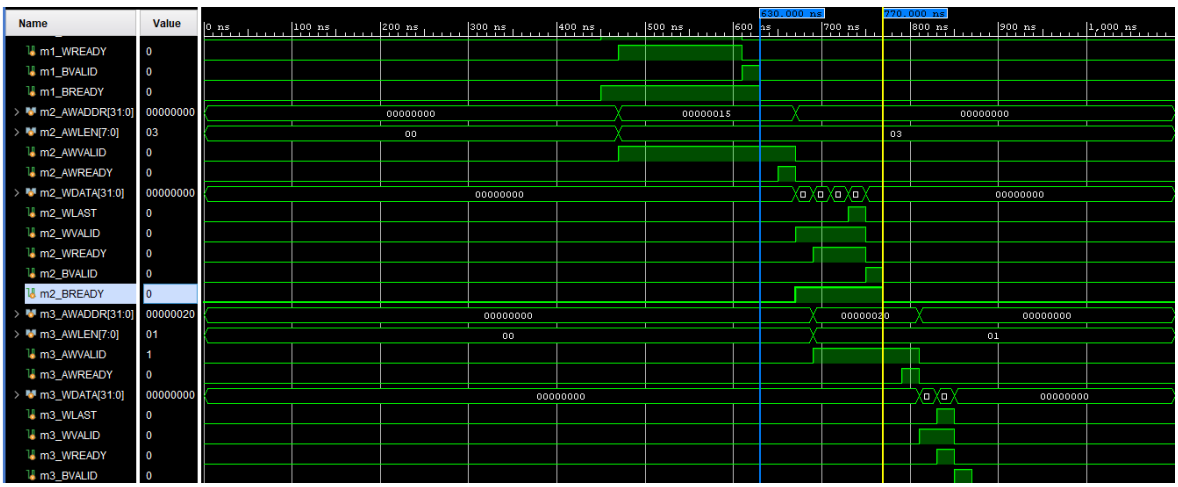

Figure 4: Write channel signals of the third and fourth processor cores without considering the arbiter.

of 0f burst lengths completed the transactions and it is the chance for the second processor core to use the shared interface to access the off-chip memory. This also means that the second processor core will wait till 410 ns to start the transaction. In the same scenario, the third and fourth processor cores will wait till 630 ns and 770 ns respectively to start the transactions as shown in Figure 4. The average waiting time for all the processor cores is 460 ns as implemented in [15]. The proposed model of this work improves the average waiting time while all the processor cores using the AXI4 interface protocol with different burst lengths by considering the arbiter architecture based on the static fixed-priority scheduling algorithm. At the program time, we assigned the high priority to the processor core that has less burst data transaction (less burst length) and low priority to the processor core that has high burst data transaction (more burst length) to improve the average waiting time for all the processor cores as shown the Figure 5–7. The write transactions for all the four-processor cores are simulated in the following simulation results. We used the burst length for each processor core in the same way as in the previous work [15]. The burst lengths are 0f, 07, 03, and 01 for the first, second, third, and fourth processor cores respectively.

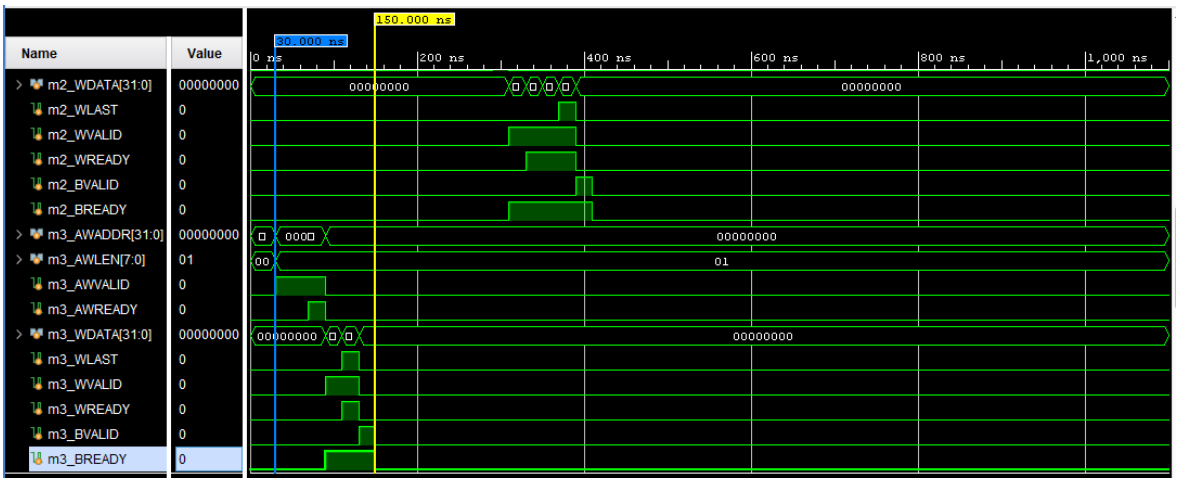

Figure 5: Write channel signals of the first and second processor cores with considering the arbiter.

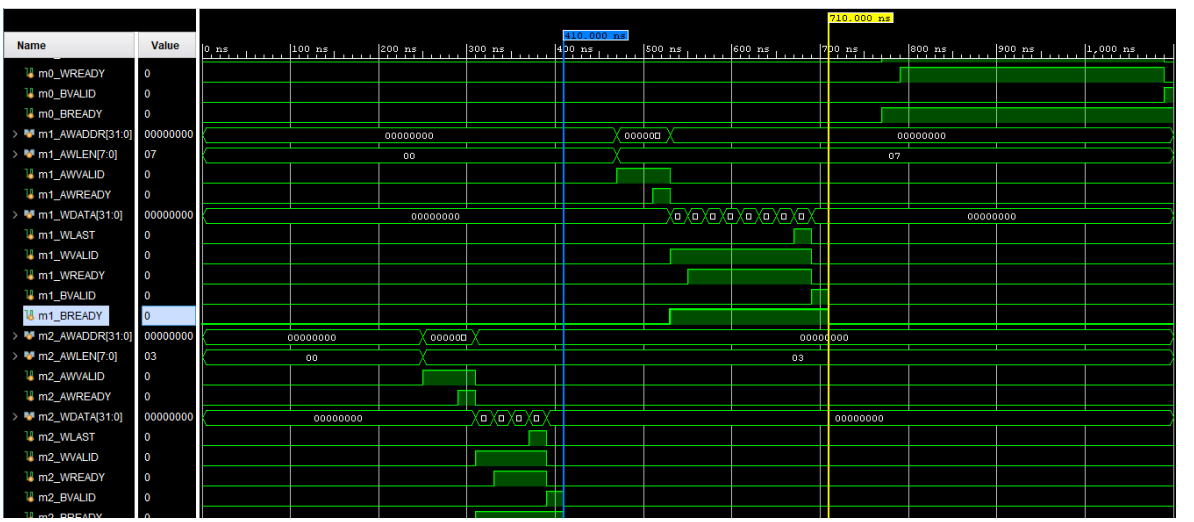

Figure 6: Write channel signals of the second and third processor cores with considering the arbiter.

In the Figure 5, we observed that the arbiter granted the fourth processor core to use the shared interface to access the off-chip memory first. The fourth processor core started the write transaction signals at the time 30 ns when the clock and reset signals are high. At the time 150 ns, the m3_BVALID and m3_BREADY signals are high which indicates that the fourth processor core with 01 burst lengths is completed and it is the chance for the third processor core to use the shared interface to access the off-chip memory because it has burst length less than the first and second processor cores. This also means that the third processor core will wait till 150 ns to start the transaction. In this scenario, the second and first processor cores will wait till 410 ns and 710 ns respectively to start the transactions as shown in Figure 6 and 7. The average waiting time for all the processor cores is 325 ns.

We observed from the simulation results that our proposed model improved the average waiting time for all the processor cores by 34.4%. The other remaining simulation results of the four-processor cores with the other different burst lengths are directly written in Table 1 and 2.

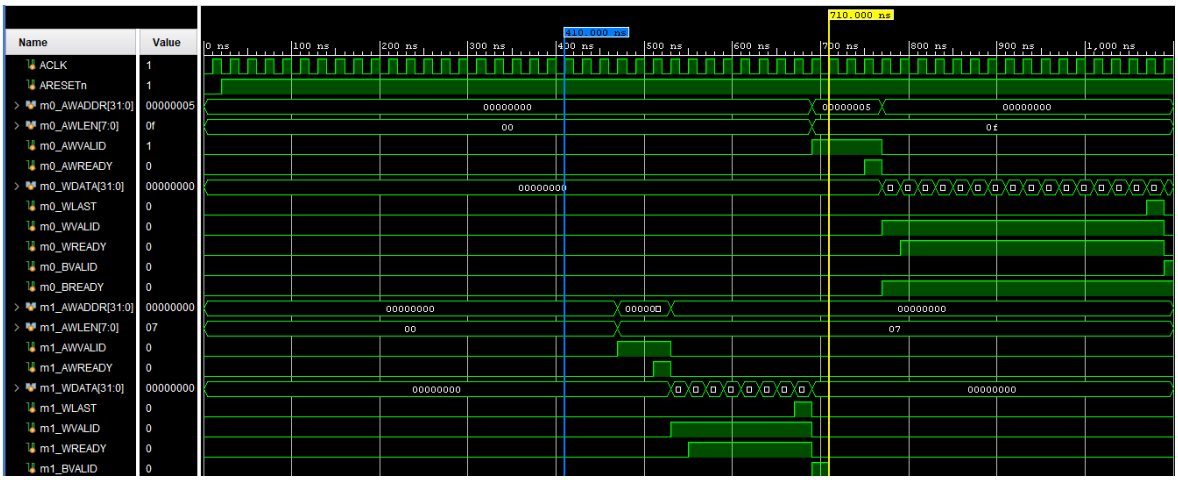

Figure 7: Write channel signals of the third and fourth processor cores with considering the arbiter.

Table 1: Waiting time for different burst lengths

	Burst Length	Waiting Time	
Master 0	1f	30 ns	30 ns
Master 1	0f	730 ns	150 ns
Master 2	07	1110 ns	410 ns
Master 3	01	1250 ns	870 ns
Average Waiting Time		780 ns	365 ns
Percentage (%)		72.5%	

Table 1 shows the other simulation results of the four-processor cores to access the off-chip memory (without/ with considering the arbiter architecture). As the data mentioned in Table 1, the four-processor cores initiate the transactions to access the off-chip memory in different burst lengths 1f, 0f, 07, and 01 (i.e. 32, 16, 8, and 2 beats of data transactions) for the first, second, third, and fourth processor cores respectively.

The column of the waiting time in Table 1 is divided into two sub-columns. The first sub-column represents the waiting time of the four-processor cores while accessing the off-chip memory without considering the arbiter architecture in the design, which means the first processor core must access first, then the second processor core, and so on regardless of the data burst length. The average waiting time for all the four-processor cores in this scenario is 780 ns. The second sub-column represents the waiting time of the four-processor cores while accessing the off-chip memory with considering the arbiter architecture based on the static fixed-priority scheduling algorithm in the design. In this scenario, only the processor core that has less burst length can access first. The average waiting time for all the four-processor cores in this scenario is 365 ns. The final percentage improvement of our proposed model of this work according to the data available in Table 1 is 72.5%.

Table 2: Waiting time for other different burst lengths

	Burst Length	Waiting Time	
Master 0	2f	30 ns	30 ns
Master 1	1f	1370 ns	190 ns
Master 2	0f	1750 ns	490 ns
Master 3	03	1970 ns	890 ns
Average Waiting Time		1280 ns	400 ns
Percentage (%)		104.8%	

Also, Table 2 shows the other simulation results of the four-processor cores initiate the transactions to access the off-chip memory in different burst lengths 2f, 1f, 0f, and 03 (i.e. 64, 32, 16, and 4 beats of data transactions) for the first, second, third, and fourth processor cores respectively.

The column of the waiting time in Table 2 is divided into two sub-columns. The first sub-column represents the waiting time of the four-processor cores while accessing the off-chip memory without considering the arbiter architecture in the design, which means the first processor core must access first, then the second processor core, and so on regardless of the data burst length. The average waiting time for

all the four-processor cores in this scenario is 1280 ns. The second sub-column is represents the waiting time of the four-processor cores while accessing the off-chip memory with considering the arbiter architecture based on the static fixed-priority scheduling algorithm in the design. In this scenario, only the processor core that has less burst length can access first. The average waiting time for all the four-processor cores in this scenario is 400 ns. The final percentage improvement of our proposed model of this work according to the data available in Table 2 is 104.8%.

The proposed design of this work is modeled in System Verilog HDL, all the simulations and synthesis are done by using the Vivado tool and FPGA ZYNQ-7 ZC702 Evaluation Board (xc7z020clg484-1).

The AXI arbiter architecture based on the static fixed-priority scheduling algorithm is easy to implement and less hardware utilization is needed as shown in the FPGA utilization of Table 3. Also, in [16], the static fixed-priority and round-robin algorithms are implemented and verified. The static fixed-priority algorithm is provides precise results and less chip area compared with the round-robin algorithm.

Table 3: FPGA Utilization summary

Logic Utilization	Available	Used	Utilization Percentage
Slice LUTs	53200	41	0.1%
Slice Registers	106400	2	0%
Slice	13300	14	0.11%
Bounded IOB	200	151	75.5%

5 Conclusions

In this paper, we have designed the AXI interconnect which includes the arbiter architecture based on the static fixed-priority scheduling algorithm. The AXI interconnect is designed to connect four-processor cores represented by the AXI masters by using the AXI4 interface protocol with different burst lengths and the off-chip memory represented by the AXI slave. The arbiter architecture based on the static fixed-priority scheduling algorithm is proposed in this work to improve the average waiting time for all the processor cores. At the program time, we have assigned the priorities for all the processor cores that depend on the data burst length for each processor core. Only the processor core that has less burst length can access the off-chip memory first as shown in the simulation figures, Tables 1 and 2 respectively. The proposed model of this work improved the existing work by 34.4% as shown in the simulation results of figures and 72.5%, and 104.8% as shown in the simulation results of Table1 and 2 respectively. The SoC design architecture is modeled in System Verilog HDL; simulation and synthesis are done by using the Vivado tool and FPGA ZYNQ-7 ZC702 Evaluation Board (xc7z020clg484-1).

Acknowledgments

This work has been supported by the Indian Council for Cultural Relations (ICCR), New-Delhi, India, and TEQIP-III official, University College of Engineering, Osmania University, Hyderabad, India.

References

[1] T. Hussain, "Memory resources aware run-time automated scheduling policy for multi-core systems," *International Journal of Microprocessors and Microsystems*, vol. 57, pp. 32–41, 2018.

[2] R. Khanam, H. Sharma, and S. Gaur "Design a less latency Arbiter for on chip communication architecture," in *International Conference on Computing, Communication and Automation*, 2015.

[3] M. N. Akhtar and O. Sidek, "An intelligent arbiter for maximum CPU utilization, fair bandwidth allocation and less latency: Survey," in *IEEE 8th International Colloquium on Signal Processing and its Applications*, 2012.

[4] M. N. Akhtar and J. M. Saleh, "Parallel adaptive arbiter for improved CPU utilization and fair bandwidth allocation," in *International Conference on Circuits, Systems, Signal Processing, Communications and Computers*, 2015.

[5] Arm Limited, "AXI Reference Guide, UG1037 (v4.0)," 2017. [Online]. Available: http://www.amba.com

[6] S. S. Math and R. B. Manjula "Design of AMBA AXI4 protocol for system-on-chip communication," *International Journal of Communication Network and Security*, vol. 1, pp. 38–42, 2012.

[7] A. Noami, A. Alahdal, B. P. Kumar, P. Chandrasekhar, and N. Safi, "High speed data transactions for

memory controller based on AXI4 interface protocol SoC," in *The IEEE 1st International Conference on Advances in Electrical, Computing, Communications and Sustainable Technologies*, 2021.

[8] J. Gupta and N. Goel, "Efficient bus arbitration protocol for SoC design," in *International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials*, 2015, pp. 396–400.

[9] A. Shrivastava and S. K. Sharma, "Various arbitration algorithm for on-chip (AMBA) shared bus multi-processor SoC," in *IEEE Students' Conference on Electrical, Electronics and Computer Science*, 2016.

[10] R. Khanam and Z. Ahmad, "Finite state machine based arbiter for on chip communication architecture," in *International Conference on Computing, Communication and Automation*, 2016.

[11] A. Ingle and P. Srividya, "Design and simulation of multi master AHB lite bus interconnect," in *Proceedings of the IEEE International Conference on Computing Methodologies and Communication*, 2017.

[12] B. M. Shankaranarayana and V. V. Malode, "Design and implementation of power efficient arbiter module for AMBA AHB protocol," *International Journal of Advanced Research in Computer Engineering & Technology*, vol. 3, pp. 2141–2145, 2014.

[13] B. Tiwari and N. Goel, "Multi-master bus interface design using efficient lottery bus arbiter," in *IEEE Annual India Conference*, 2016.

[14] A. Noami, B. P. Kumar, and P. Chandrasekhar "Design and implementation of a united multi-core memory controller using AXI4-lite interface protocol," *International Journal on Emerging Technologies*, vol. 11, no. 3, pp. 468–475, 2020.

[15] A. Noami, B. P. Kumar, and P. Chandrasekhar, "High performance AXI4 interface protocol for multi-core memory controller on SoC," in *the Springer 4th International Conference on Data Engineering and Communication Technology*, 2020.

[16] S. V. Vijayalakshmi, A. Apsara, K. Preetha, and S. Cammillus "Memory arbitration in DDR3," *International Journal of Recent Technology and Engineering*, vol. 8, pp. 3344–3347, 2020.