

Implementation Of An Adaptive-Dynamic Arbitration Scheme For The Multilayer AHB Busmatrix

Dr. Fazal Noorbasha*, B.Srinivas**, Venkata Aravind Bezawada***,
V.Sai Praveen ****

VLSI Research & Development Group, Department of ECE, KLU University, Guntur, AP-522502 INDIA

ABSTRACT:

In this paper, the adaptive dynamic arbitration scheme is being implemented on the slave side arbitration based on AMBA AHB protocol. The multilayered advanced high-performance bus (ML-AHB) bus matrix is an interconnection between multiple masters and multiple slaves in a system. The design and implementation of a flexible arbiter for the ML-AHB bus matrix is to support three priority policies—fixed priority, round robin, and dynamic priority and three data multiplexing modes—transfer, transaction, and desired transfer length. The slave side arbiter dynamically selects one of the nine possible arbitration schemes based upon the priority-level notifications and the desired transfer length from the masters so that arbitration leads to the maximum performance. The area overhead of the adaptive dynamic arbitration scheme will be larger than those of the other arbitration schemes and improves the throughput when compared to other schemes. Among the nine arbitration schemes, the adaptive dynamic arbitration scheme is the efficient one and the master which has accessed the bandwidth less number of times will be given highest priority and will get the grant signals.

Keywords: *ML-AHB bus matrix, system-on-a-chip (SoC), Adaptive dynamic arbitration, slave side arbitration, high performance*

1. INTRODUCTION

The on-chip bus plays a key role in the system-on-a-chip (SoC) design by enabling the efficient integration of heterogeneous system components such as CPUs, DSPs, application-specific cores, memories, and custom logic. Recently, as the level of design complexity has become higher, SoC designs require a system bus with high bandwidth to perform multiple operations in parallel [1]. To solve the bandwidth problems, there have been several types of high-performance on-chip buses proposed, such as the multilayer AHB (ML-AHB) busmatrix from ARM, the PLB crossbar switch from IBM, and CONMAX from Silicore. Among them, the ML-AHB busmatrix has been widely used in many SoC designs. This is because

of the simplicity of the AMBA bus of ARM, which attracts many IP designers, and the good architecture of the AMBA bus for applying embedded systems with low power. The multilayered advanced high-performance bus (ML-AHB) bus matrix is an interconnection between multiple masters and multiple slaves in a system. The master and the slave communicate in terms of request and grant signals. The master merely starts a burst transaction and waits for the slave response to proceed to the next transfer. However, the ML-AHB busmatrix of ARM offers only transfer-based fixed-priority and round-robin arbitration schemes. In fixed priority arbitration scheme, each master is assigned a fixed priority value. It is simple in implementation and has small area cost. But in heavy communication traffic, master that has low priority value cannot get a grant signal. In round robin arbitration scheme, each master is allotted a fixed time slot. If the new master sends a request in between, then that master has to wait until all masters complete their tasks. This is achieved by using a more complex interconnection matrix and gives the benefit of both increased overall bus bandwidth and a more flexible system structure. In particular, the ML-AHB busmatrix uses slave-side arbitration. Slave-side arbitration is different from master-side arbitration in terms of request and grant signals since, in the former, the master merely starts a burst transaction and waits for the slave response to proceed to the next transfer. Therefore, the unit of arbitration can be a transaction or a transfer. The transaction-based arbiter multiplexes the data transfer based on the burst transaction, and the transfer-based arbiter switches the data transfer based on a single transfer. However, the ML-AHB busmatrix of ARM presents only transfer-based arbitration schemes, i.e., transfer based fixed-priority and round-robin arbitration schemes. This limitation on the arbitration scheme may lead to degradation of the system performance because the arbitration scheme is usually dependent on the application requirements; recent applications are likewise becoming more complex and diverse [2]. By implementing an efficient arbitration scheme, the system performance can be tuned to better suit

applications. For a high-performance on-chip bus, several studies related to the arbitration scheme have been proposed, such as table-lookup-based crossbar arbitration, two-level time-division multiplexing (TDM) scheduling, token-ring mechanism, dynamic bus distribution algorithm, and LOTTERYBUS. However, these approaches employ master-side arbitration. Therefore, they can only control priority policy and also present some limitations when handling the transfer-based arbitration scheme since master-side arbitration uses a centralized arbiter. In contrast, it is possible to deal with the transfer-based arbitration scheme as well as the transaction-based arbitration scheme in slave-side arbitration. In this paper, we propose a flexible arbiter based on the adaptive-dynamic (AD) arbitration scheme for the ML-AHB busmatrix [3].

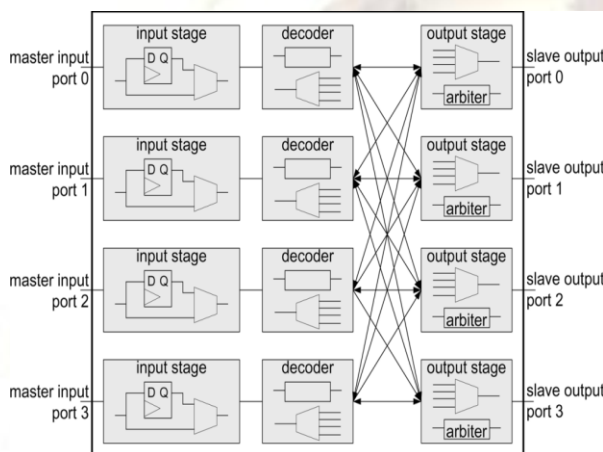


Fig.1. Overall structure of the ML-AHB busmatrix of ARM.

In Section II, we briefly explain the arbitration schemes for the ML-AHB busmatrix of ARM, while Section III describes an implementation method for our flexible arbiter based upon the AD arbitration scheme for the ML-AHB busmatrix. We then present implementation results and performance analysis in Section IV, simulation results in Section V and concluding remarks in Section VI.

II. ARBITRATION SCHEMES FOR THE ML- AHB BUSMATRIX OF ARM

The ML-AHB busmatrix of ARM consists of the input stage, decoder, and output stage, including an arbiter. Fig. 1 shows the overall structure of the ML-AHB busmatrix of ARM. The input stage is responsible for holding the address and control information when transfer to a slave is not able to commence immediately. The decoder determines which slave that a transfer is destined for. The output stage is used to select which of the various master input ports is routed to the slave. Each output stage has an arbiter. The arbiter determines which input stage has to perform a

transfer to the slave and decides which the highest priority is currently. The ML-AHB busmatrix employs slave-side arbitration, in which the arbiters are located in front of each slave port, as shown in Fig. 1. The master simply starts a transaction and waits for the slave response to proceed to the next transfer. Therefore, the unit of arbitration can be a transaction or a transfer. However, the ML-AHB busmatrix of ARM furnishes only transfer-based arbitration schemes, specifically transfer-based fixed-priority and round-robin arbitration schemes. The transfer-based fixed-priority (round-robin) arbiter multiplexes the data transfer based on a single transfer in a fixed-priority or round-robin fashion [4-7].

III. AD ARBITRATION SCHEME FOR THE ML- AHB BUSMATRIX

An assumption is made that the masters can change their priority level and can issue the desired transfer length to the arbiters in order to implement an AD arbitration scheme. This assumption should be valid because the system developer generally recognizes the features of the target applications. For example, some masters in embedded systems are required to complete their job for given timing constraints, resulting in the satisfaction of system-level timing constraints. The computation time of each master is predictable, but it is not easy to foresee the data transfer time since the on-chip bus is usually shared by several masters [7-9].

Our AD arbitration scheme has the following advantages:

- 1) It can adjust the processed data unit;
- 2) it changes the priority policies during runtime; and
- 3) it is easy to tune the arbitration scheme according to the characteristics of the target application.

Hence, our arbiter is able to not only deal with the transfer-based fixed-priority, round-robin, and dynamic-priority arbitration schemes but also manage the transaction-based fixed-priority, round-robin, and dynamic-priority arbitration schemes. Furthermore, our arbiter provides the desired-transfer-length-based fixed-priority, round-robin, and dynamic-priority arbitration schemes. In addition, the proposed AD arbiter selects one of the nine possible arbitration schemes based on the priority-level notifications and the desired transfer length from the masters to ensure that the arbitration leads to the maximum performance [10].

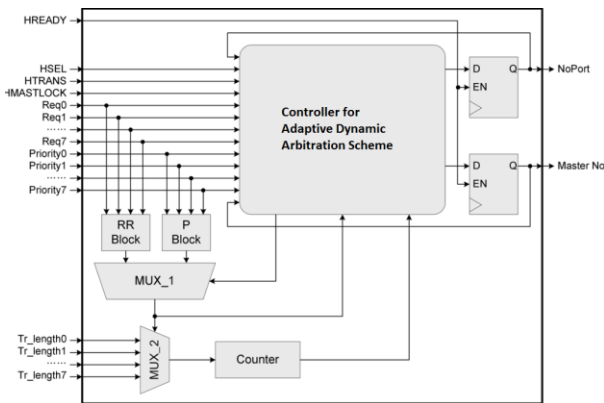


Fig.2. Internal structure of our arbiter.

Fig.2 shows the internal structure of our arbiter based upon the AD arbitration scheme. the NoPort signal means that none of the masters must be selected and that the address and control signals to the shared slave must be driven to an inactive state, while Master No. indicates the currently selected master number generated by the controller for the AD arbitration scheme. In general, our arbiter consists of an RR block, a P block, two multiplexers, a counter, a controller, and two flip-flops. MUX_1 and MUX_2 are used to select the arbitration scheme and the desired transfer length of a master, respectively. A counter calculates the transfer length, with two flip-flops being inserted to avoid the attempts by the critical path to arbitrate. An RR block (P block) performs the round-robin- or priority-based arbitration scheme. Fig. 5 shows the internal process of an RR block.

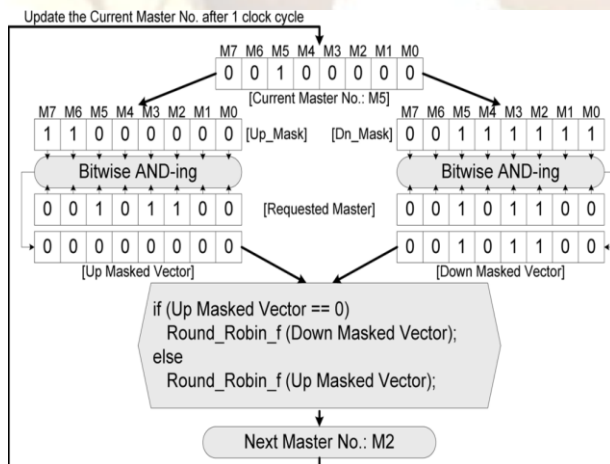


Fig.3. Internal process of the RR block.

Initially, we create the up- and down-mask vectors (Up_Mask and Dn_Mask) based on the number of currently selected masters, as shown in Fig.3. We then generate the up- or down-masked vector created through bitwise AND-ing operation between the mask vector and the requested master vector.

After generating the up- and down-masked vectors, we examine each masked vector as to whether they are zero or not. If the up-masked vector is zero, the down-masked vector is inserted to the input parameter of the round-robin function; if it is not zero, the up-masked vector is the one inserted. A master for the next transfer is chosen by the round-robin function, and the current master is updated after 1 clock cycle. The RR block is then performed by repeating the arbitration procedure shown in Fig.3.

A master for the next transfer is selected, with the priority level of the least significant bit in Masked_Vector being the highest. If we modify the range of Masked_Vector to "0 to Masked_Vector'left," then the priority level of the most significant bit in Masked_Vector becomes the highest.

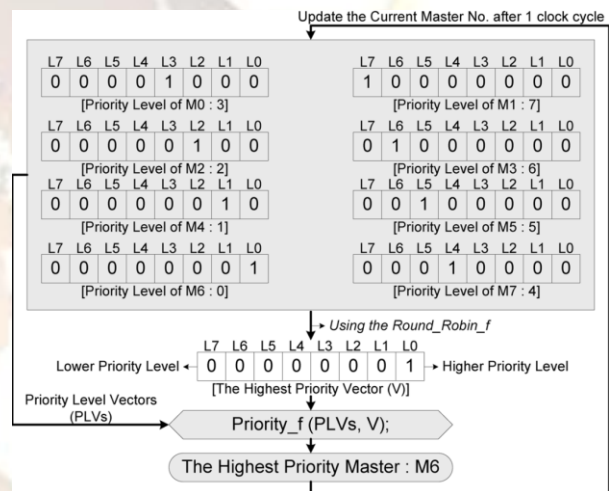


Fig.4. Internal procedure of the P block.

Fig.4 shows the internal procedure of the P block. First of all, we create the highest priority vector (V) through the round-robin function. After generating the highest priority vector (V), the priority-level vectors and the highest priority vector (V) are inserted to the input parameters of the priority function. The master with the highest priority is chosen by the priority function, while the current master is updated after 1 clock cycle. The master with the highest priority is selected in Fig4 [11].

A controller compares the priority levels of the requesting masters. If the masters have equal priorities, the controller selects the round-robin arbitration scheme (RR block); in other cases, it chooses the priority arbitration scheme (P block). The controller also makes the final decision on the master for the next transfer based on the transfer length of the selected master. The control process follows the following three steps.

- 1) If HMASTLOCK is asserted, the same master remains selected.

2) If HMASTLOCK is not asserted and the currently selected master does not exist, the following hold.

a) If no master is requesting access, the NoPort signal is asserted.

b) Otherwise, a new master for the next transfer is initially selected. If the masters have equal priorities, the round-robin arbitration scheme is selected; otherwise, the priority arbitration scheme is chosen. In addition, the counter is updated based on the transfer length of the selected master.

3) If none of the previous statements applies, the following hold.

a) If the counter is expired, the following hold.

i) If the requesting masters do not exist, the No-Port signal is updated based on the HSEL signal of the currently selected master. If the HSEL signal is "1," the same master remains selected, and the NoPort signal is deasserted. Otherwise, the NoPort signal is asserted.

ii) Otherwise, a master for the next transfer is selected based on the priority levels of the requesting masters. Also, the counter is updated.

b) If the counter is not expired, and the HSEL signal of the current master is "1," the same master remains selected, and the counter is decreased.

c) If the currently selected master completes a transaction before the counter is expired, the following hold.

i) If the requesting masters do not exist, the No-Port signal is asserted.

ii) Otherwise, a master for the next transfer is chosen based on the priority levels of the requesting masters, and the counter is updated [12].

The AD arbitration scheme is achieved through iteration of the aforementioned steps. Combining the priority level and the desired transfer length of the masters allows our arbiter to handle the transfer-based fixed-priority, round-robin, and dynamic-priority arbitration schemes (abbreviated as the FT, RT, and DT arbitration schemes, respectively), as well as the transaction-based fixed-priority, round-robin, and dynamic-priority arbitration schemes (abbreviated as the FR, RR, and DR arbitration schemes, respectively). Moreover, our arbiter can also deal with the desired-transfer-length-based fixed-priority, round-robin, and dynamic-priority arbitration schemes (abbreviated as the FL, RL, and DL arbitration schemes, respectively). The transfer or transaction-based arbiter switches the data transfer based upon a single transfer (burst transaction), and the desired-transfer-length-based arbiter multiplexes the data transfer based on the transfer length assigned by the masters [13-15].

IV. IMPLEMENTATION RESULTS AND PERFORMANCE ANALYSIS

We implemented different slave-side arbitration schemes for the ML-AHB busmatrix. Each arbitration-scheme-based busmatrix was implemented with synthesizable RTL Verilog targeting XILINX FPGA (XC3S200). The XILINX design tool (ISE 12.1i) was used to measure the total area. The implemented arbitration schemes are as follows:

- FT, FR, RT, RR, DT, DR, and AD arbitration schemes.

The total area of the AD-based busmatrix is 9%–25% larger than those of the other busmatrixes. This may be due to our AD-based busmatrix also requiring the comparator to compare the priority of the masters and the counters to calculate the transfer length. Although our AD-based busmatrix occupies more area than the other busmatrixes, our arbiter is able to deal with varied arbitration schemes such as the FT, FR, RT, RR, DT, and DR arbitration schemes.

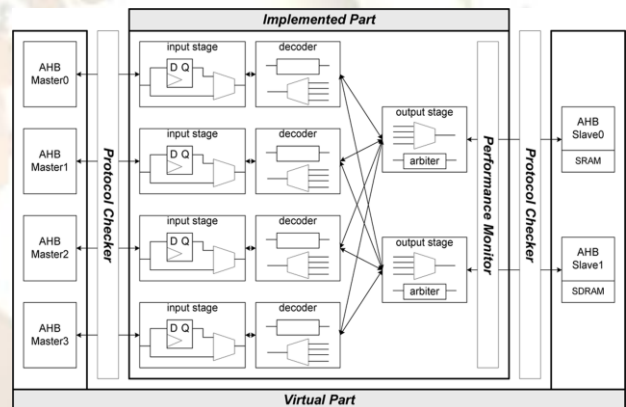


Fig.5. Simulation environment for performance analysis

Fig.5 shows our simulation environment.

In our simulation environment, the clock frequencies of all components are 100 MHz (10 ns). The implemented ML-AHB busmatrix has a 32-b address bus, a 32-b write data bus, a 32-b read data bus, a 15-b control bus, and a 3-b response bus. Meanwhile, the simulation environment consists of both an implemented and a virtual part. The former corresponds to the ML-AHB busmatrixes with different arbitration schemes and consists of four masters and two slaves. Specifically, we only considered two target slaves, which is when conflict frequently happens. The masters then access these in order to focus on the performance analysis based on the arbitration schemes of each busmatrix. The virtual part, however, is composed of AHB masters and AHB slaves. The AHB master generates the transactions, with the transactions of the masters

having the same length as an 8-beat incrementing burst type. The AHB slave responds to the transfers of the masters. Both the AHB masters and slaves are fully compatible with the AMBA AHB protocol. For a more realistic model of a SoC design, we modeled the AHB masters after the features of the processor and DMA with verilog at the behavioral level. For the AHB slaves, we used the real SRAM, SDRAM, and SDRAM controller RTL models used in many applications. We also constructed the protocol checker and performance monitor modules with the verilog and foreign language interface (FLI C module) to ensure the reliability of our performance simulations. Prior to the simulation, the workloads should be determined as they affect the simulation results. However, determining the appropriate workloads of real applications is difficult because these can only be obtained when all applications with real input data are specifically modeled. Instead, the workloads for performance simulation are obtained through synthetic workload generation with the following parameters.

- 1) The distribution of transactions. This indicates what proportion of the total transactions that each master is responsible for.
- 2) The ratio of the nonbus transaction time to the total transaction time per AHB master, where the total transaction time consists of a nonbus transaction (internal transaction of the master) time and a bus transaction (external transaction of the master through the busmatrix) time.
- 3) The latency time of the accessed slave by each master. These parameters determine the delay of components in the virtual part. Through synthetic workload generation, various possible situations are investigated, where the ML-AHB busmatrixes with each arbitration scheme can be utilized well. In this regard, we found three useful categories of experiments to identify the effects of the following factors:
 - 1) job length of the masters;
 - 2) latency time of the slaves;
 - 3) both the job length of the masters and the latency time of the slaves.

The dynamic-priority-based arbitration scheme has the advantage for throughput when there are few masters with long job lengths in a system; in other cases, the round-robin-based arbitration scheme can get higher throughput than other arbitration schemes. In addition, the arbitration scheme with transaction-based multiplexing performs better than the same arbitration scheme with single-transfer-based switching in applications with frequent access to long-latency slaves such as SDRAM. The slave for the first category is the

SRAM-type AHB slave (AHB slave0 in Fig. 10) without latency for access, while the slave for the second category is the SDRAM-type AHB slave with a long latency time for access. The slave for the third category can be an AHB slave0 or an AHB slave1. In particular, the target addresses are generated based on the uniform distribution random number function between AHB slave0 and AHB slave1. Therefore, each master communicates with the slaves with the same probability in the third category. We performed a number of performance simulations at various job lengths and observed no difference in the results of the performance simulation at specific job lengths. The specific job length was 4800, and we decided the job length for performance analysis to be the same at 4800. In addition, this job length explicitly exhibits the features of each arbitration scheme very well.

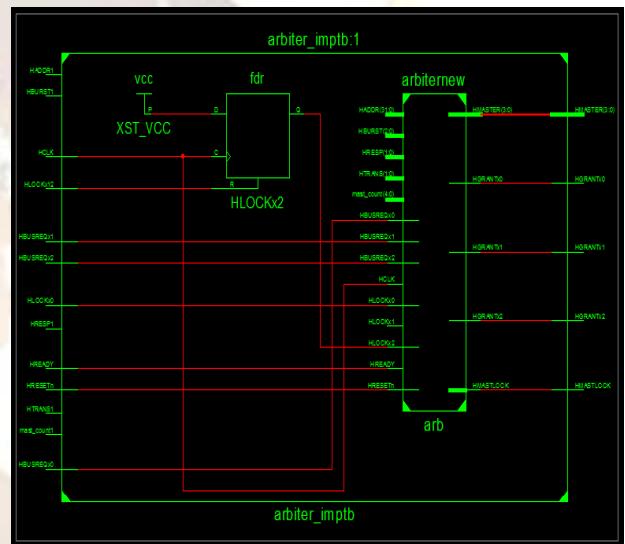


Fig 6: RTL Schematic of AD Arbiter

Fig 6 represents the RTL Schematic of AD Arbiter. In this block the implemented ML-AHB busmatrix has a 32-b address bus, a 32-b write data bus, a 32-b read data bus, a 15-b control bus, and a 3-b response bus.

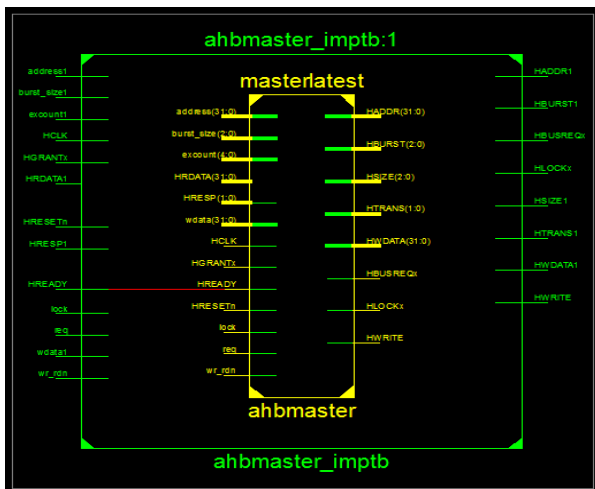


Fig 7: RTL Schematic of AHB Master

Fig 7 represents RTL Schematic of AHB Master

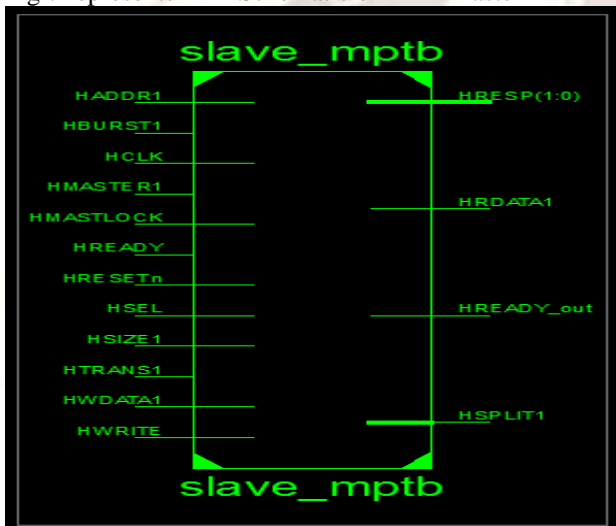
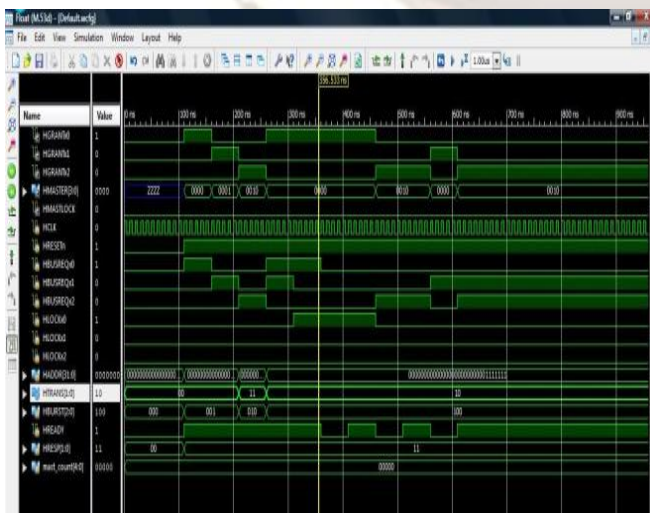


Fig 8: RTL Schematic of AHB Slave

V.SIMULATION RESULTS



VI.CONCLUSION

In this paper, we proposed a flexible arbiter based on the AD arbitration scheme for the ML-AHB busmatrix. Our arbiter supports three priority policies-fixed priority, round-robin, and dynamic priority-and three approaches to data multiplexing-transfer, transaction, and desired transfer length; in other words, there are nine possible arbitration schemes. In addition, the proposed AD arbiter selects one of the nine possible arbitration schemes based on the priority-level notifications and the desired transfer length from the masters to allow the arbitration to lead to the maximum performance. Experimental results show that, although the area of the proposed AD arbitration scheme is 9%–25% larger than those of other arbitration schemes, our arbiter improves the throughput by 14%–62% compared with other schemes. We therefore expect that it would be better to apply our AD arbitration scheme to an application-specific system because it is easy to tune the arbitration scheme according to the features of the target system.

REFERENCES

- [1] M. Drinic, D. Kirovski, S. Megerian, and M. Potkonjak, "Latencyguided on-chip bus-network design," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 25, no. 12, pp. 2663–2673, Dec. 2006.
- [2] S. Y. Hwang, K. S. Jhang, H. J. Park, Y. H. Bae, and H. J. Cho, "An ameliorated design method of ML-AHB busmatrix," ETRI J., vol. 28, no. 3, pp. 397–400, Jun. 2006.
- [3] ARM, "AHB Example AMBA System," 2001 [Online]. Available: http://www.arm.com/products/solutions/AMBA_Spec.html
- [4] IBM, New York, "32-bit Processor Local Bus Architecture Specification," 2001.
- [5] R. Usselmann, "WISHBONE interconnect matrix IP core," Open-Cores, 2002. [Online]. Available: http://www.opencores.org/?do=project=wb_conmax
- [6] N.-J. Kim and H.-J. Lee, "Design of AMBA wrappers for multiplexed operations," in Proc. Int. Conf. ICCAS, Jun. 2004, vol. 2, pp. 1438–1442.
- [7] D. Flynn, "AMBA: Enabling reusable on-chip designs," IEEE Micro, vol. 17, no. 4, pp. 20–27, Jul./Aug. 1997.
- [8] S. Y. Hwang, H.-J. Park, and K.-S. Jhang, "Performance analysis of slave-side arbitration schemes for the multi-layer AHB busmatrix," J. KISS, Comput. Syst. Theory, vol. 34, no. 5, pp. 257–266, Jun. 2007.

- [9] S. S. Kallakuri and A. Doboli, "Customization of arbitration policies and buffer space distribution using continuous-time Markov decision processes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 2, pp. 240–245, Feb. 2007.
- [10] D. Seo and M. Thottethodi, "Table-lookup based crossbar arbitration for minimal-routed, 2D mesh and torus networks," in *Proc. Int. Conf. IPDPS*, Mar. 2007, pp. 1–10.
- [11] K. Lahiri, A. Raghunathan, and S. Dey, "Performance analysis of systems with multi-channel communication architectures," in *Proc. Int. Conf. VLSI Design*, Jan. 2000, pp. 530–537.
- [12] J. Turner and N. Yamanaka, "Architectural choices in large scale ATM switches," *IEICE Trans. Commun.*, vol. E-81B, no. 2, pp. 120–137, Feb. 1998.
- [13] C. H. Pyoun, C. H. Lin, H. S. Kim, and J. W. Chong, "The efficient bus arbitration scheme in SoC environment," in *Proc. Int. Conf. SoC Real-Time Appl.*, Jul. 2003, pp. 311–315.
- [14] K. Lahiri, A. Raghunathan, and G. Lakshminarayana, "The LOTTERYBUS on-chip communication architecture," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 6, pp. 596–608, Jun. 2006.
- [15] S. Y. Hwang, H. J. Park, and K. S. Jhang, "An Efficient Implementation Method of Arbiter for the ML-AHB Busmatrix." Berlin, Germany: Springer-Verlag, May 2007, vol. 4523, LNCS, pp. 229–240.



Dr. Fazal Noorbasha was born on 29th April 1982. He received his, B.Sc. Degree in Electronics Sciences from BCAS College, Bapatla, Guntur, A.P., Affiliated to the Acharya Nagarjuna University, Guntur, Andhra Pradesh, India, in 2003, M.Sc. Degree in Electronics Sciences from the Dr. HariSingh Gour University, Sagar, Madhya Pradesh, India, in 2006, M.Tech. Degree in VLSI Technology, from the North Maharashtra University, Jalgaon, Maharashtra, INDIA in 2008, and Ph.D. Degree in VLSI from Department Of Physics and Electronics, Dr. HariSingh Gour Central University, Sagar, Madhya Pradesh, India, in 2011. Presently he is working as a Assistant Professor, Department of Electronics and Communication Engineering, KL University,

Guntur, Andhra Pradesh, India, where he has been engaged in teaching, research and development of Low-power, High-speed CMOS VLSI SoC, Memory Processors LSI's, Fault Testing in VLSI, Embedded Systems and Nanotechnology. He is a Scientific and Technical Committee & Editorial Review Board Member in Engineering and Applied Sciences of World Academy of Science Engineering and Technology (WASET), Advisory Board Member of International Journal of Advances Engineering & Technology (IJAET), Member of International Association of Engineers (IAENG) and Senior Member of International Association of Computer Science and Information Technology (IACSIT). He has published over 20 Science and Technical papers in various International and National reputed journals and conferences.



Srinivas Boddu was born in A.P,India. He received the B.Tech degree in Electronics&communications engineering from Jawaharlal Nehru Technological University in 2009. Presently he is pursuing M.Tech VLSI Design in KL University. His research interests include FPGA Implementation, Low Power Design.

Venkata Aravind Bezawada was born in A.P,India. He received the **B.Tech** degree in **Electronics& communications Engineering** from Jawaharlal Nehru Technological University in 2009. Presently he is pursuing M.Tech VLSI Design in KL University. His research interests include Physical Design, Low Power Design.



Sai Praveen Venigalla was born in A.P, India. He received the B.Tech degree in **Electronics&communication Engineering** from Jawaharlal Nehru Technological University in 2009. Presently he is pursuing M.Tech VLSI Design in KL University. His research interests include FPGA Implementation, Low Power Design.