Flexible Arbiter Based on Dynamic Arbitration Scheme for Ml-Ahb Busmatrix

Manjunath M., Sunitha S. L., B. N. Shobha

Abstract— The multi-layer advanced high-performance bus (ML-AHB) matrix proposed by ARM is an excellent architecture for applying embedded systems with low power. However, there is one clock cycle delay for each master in the ML-AHB Bus Matrix of the advanced microcontroller bus architecture (AMBA) design kit (ADK) whenever a master starts new transactions or changes the slave layers. The total area and power consumption of the ML-AHB Bus Matrix of an ADK is increases due to the heavy input stages. Due to heavy input stage the cost of arbitration scheme becomes high. 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. Previous works solved this issue by minimizing the latencies of several latency-critical masters, but a side effect of these methods is that they can increase the latencies of other masters; hence, they may violate the given timing constraints. It is better to apply improved Bus Matrix to some applications that do not require the time division multiple accesses to the slaves. This paper adapts improved ML-AHB Bus Matrix to multimedia applications such as a video phone, MPEG-4, and H.264 codec and presents flexible arbiter based on the Dynamic Arbitration scheme for the ML-AHB bus matrix. The arbiter supports three priority policies-fixed priority, round-robin, and dynamic priority-and three approaches to data multiplexing-transfer, transaction, and desired transfer length. Experimental results show that, although the area of the proposed Dynamic Arbitration scheme is 9%–25% larger than those of other arbitration schemes, the arbiter scheme improves the throughput by 14%–62% compared with other schemes.

Index Terms—Multilayer AHB (ML-AHB) bus matrix, on-chip bus, Dynamic arbitration scheme, slave side arbitration, system-on-a-chip (SoC).

I. INTRODUCTION

The ON-CHIP bus plays a key role in the system-on-chip (SoC) design by enabling the efficient integration of heterogeneous system components such as CPUs, DSPs, application-specific cores, memories, and custom logic.

"As the level of design complexity has become higher, SoC designs require a system bus with high bandwidth to perform multiple operations in parallel. Several types of high-performance on-chip buses proposed, such as the ML-AHB bus matrix 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, because of the simplicity of the bus.

On-chip communication architectures play an important role in determining the overall performance of SoC designs. Communication architectures should be flexible so as to offer high performance over a wide range of traffic characteristics. In particular, the resource sharing mechanism of the communication architecture, which determines how the often conflicting requirements of different components are served, is of utmost importance. Conventional SoC architectures typically employ priority or TDMA based communication architectures. However, these techniques are often inadequate. In the former, low-priority components may suffer from starvation, while in the latter, depending on the request profile, high-priority traffic may be subject to large latencies.

The multi-layer AHB busmatrix proposed by ARM is a highly efficient on chip bus that allows parallel access paths between multiple masters and slaves in a system. However, the ML-AHB busmatrix of ARM offers only transfer-based fixed-priority and round-robin arbitration schemes. This work presents a way to improve the arbiter implementation of the ML-AHB busmatrix. The proposed arbiter, which is Dynamic Arbiter , selects one of the nine possible arbitration schemes based upon the priority-level and the desired transfer length from the masters so that arbitration leads to the maximum performance. Dynamic 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.

A. MULTILAYER AHB BUS

The primary objective of this work is to provide an effective means of bandwidth. To solve the bandwidth problems, there have been several types of high-performance on-chip buses proposed, such as the ML-AHB bus matrix from ARM. In slave side arbitration it is possible to deal with transfer-based arbitration scheme as well as transaction-based arbitration scheme. This work used a flexible arbiter based on Dynamic Arbitration scheme for ML-AHB bus matrix.

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 a Dynamic arbitration scheme. This assumption should be valid because the system developer generally recognizes the features of the target applications. In this flexible 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 this flexible arbiter provides the desired-transfer-length-based fixed-priority, round-robin and dynamic-priority arbitration schemes.

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.

Manuscript received September 02, 2012.

Manjunath M, ECE, Visvesvaraya Technological University, Mandya, India

Dr. Sunitha S.L., ECE, Visvesvaraya Technological University, Mandya, India

B.N. Shobha, ECE, Visvesvaraya Technological University, Banglore, India

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.

B. METHODOLOGY

The ML-AHB bus matrix of ARM provides only two arbitration schemes. FT and RT arbitration schemes, thus, we compared the FT- and RT-based bus matrixes of ARM with our corresponding bus matrixes in the area overhead to show the credibility of this implementation. The total areas of FT- and RT-based bus matrixes decreased by 21% and 13% on average, respectively, compared with the FT- and RT-based bus matrixes of ARM. The reason is that, this work adapted the bit masking mechanism to our bus matrixes to reduce the area of the arbiter, while ARM used multiple priority encoders, a multiplexer, and a demultiplexer to implement the arbiters of the bus matrixes. This Dynamic Arbiter based bus matrix also requiring the comparator to compare the priority of the masters and the counters to calculate the transfer length. Although this AM- based bus matrix occupies more area than the other bus matrixes, this arbiter is able to deal with varied arbitration schemes such as the FT, FR, RT, RR, DT, and DR arbitration schemes.

II. SYNTHESIS OF ARBITRATION SCHEME

In simulation environment, the clock frequencies of all components are 100 MHz (10 ns). The implemented ML-AHB bus matrix 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 optimization and implemented part. The former corresponds to the ML-AHB bus matrixes with different arbitration schemes and consists of four masters and two slaves. Specifically, this work considered four 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 bus matrix. The transactions of the masters having the same length as an 8-bit incrementing burst type.

The proposed arbiter, which is automated scheme, selects one of the nine possible arbitration schemes based upon the priority-level and the desired transfer length from the masters so that arbitration leads to the maximum performance. In this paper, we proposed a flexible arbiter based on the SM arbitration scheme for the ML-AHB bus matrix. This 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 Dynamic 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. In Dynamic Arbitration scheme can keep the latency close to its given constraint by adjusting the priority level and transfer length of the masters.

In this work, Dynamic Arbitration scheme is an improved design method to remove the one clock cycle delay in the ML-AHB Bus Matrix of an ADK. And also remarkably reduce the total area and power consumption of the ML-AHB Bus Matrix of an ADK with the elimination of the heavy input stages. With the removal of the input stage, but at the cost of some restrictions on the arbitration scheme, it can take away the one clock cycle delay and reduce the total area, clock period, average static power, and dynamic power consumption by 33%, 33%, 64% and 42%, respectively, compared with those of the Bus Matrix of an ADK.

III. SIMULATION RESULTS

Fig.1 shows the configurations for the fixed-priority arbitration schemes. In this figure, the smaller the priority level number, the higher the priority level. In the fixed-priority arbitration schemes, each master has a static priority. In transfer-based arbitration, however, the transfer length is allocated as 1, indicating a single transfer; in transaction-based arbitration, the transfer length is equal to the HBURST signal, which refers to the transaction type (transfer). In addition, the transfer length for the desired-transfer-length-based arbitration is allotted by the demand of each master. ("#" indicates the transfer number).The arbitration results are as follows

1)FTarbitrationscheme:M2(#0),M2(#1),M2(#2),M1(#0),M1(#1),M1(#2),M1(#3),M1(#4),M0(#0),M0(#1),M0(#2),M0(#3),M0(#4),M0(#5),M0(#6),M0(#7),M1(#5),M1(#6),M1(#7), M2(#3), M2(#4), M2(#5),M2(#6), M2(#7), M3(#0), M3(#1), M3(#2), M3(#3),M3(#4), M3(#5), M3(#6), M3(#7).

2) FR arbitration scheme: M2 (#0), M2 (#1), M2 (#2), M2 (#3),
M2(#4), M2(#5), M2(#6), M2(#7), M0(#0), M0(#1),M0(#2), M0(#3), M0(#4), M0(#5), M0(#6), M0(#7),M1(#0), M1(#1), M1(#2), M1(#3), M1(#4), M1(#5), M1(#6), M1(#7),M3(#0), M3(#1), M3(#2), M3(#3),M3(#4), M3(#5), M3(#6), M3(#7).

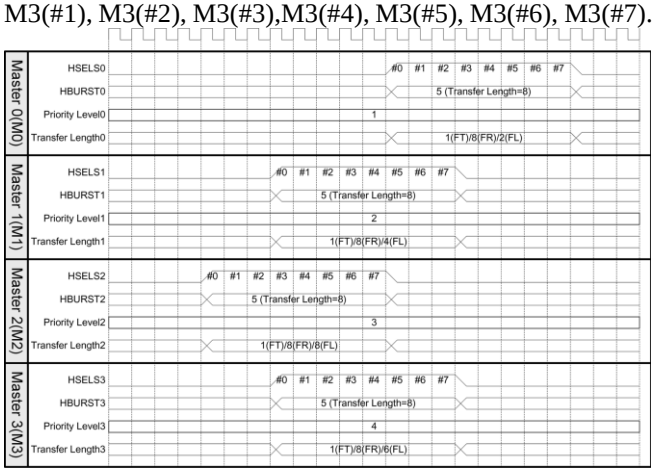

Fig. 1. Configuration for the fixed priority

3) FL arbitration scheme: M2(#0), M2(#1), M2(#2), M2(#3), M2(#4), M2(#5), M2(#6), M2(#7), M0(#0), M0(#1),M0(#2), M0(#3), M0(#4), M0(#5), M0(#6), M0(#7),M1(#0), M1(#1), M1(#2), M1(#3), M1(#4), M1(#5),M1(#6), M1(#7), M3(#0), M3(#1), M3(#2), M3(#3),M3(#4), M3(#5), M3(#6), M3(#7).

Round-robin (RR) is one of the simplest scheduling algorithms for processes in an operating system. As the term is generally used, time slices are assigned to each process in equal portions and in circular order, handling all processes without priority (also known as cyclic executive). Round-robin scheduling is simple, easy to implement, and starvation-free.

Round-robin scheduling can also be applied to other scheduling problems, such as data packet scheduling in computer networks. In this case, the result of transaction-based arbitration is equal to that of desired-transfer-length-based arbitration because the priority levels of all the masters are fixed. Fig.2 shows the combinations for the round-robin arbitration schemes. In these schemes, the masters have equal priorities, with the transfer length being assigned as 1 in transfer-based arbitration and 8 in transaction-based arbitration. Also, in desired-transfer length- based arbitration, the transfer length is assigned by the demand of each master. The arbitration results are as follows.

1) RT arbitration scheme: M0(#0), M1(#0), M2(#0), M3(#0), M0(#1), M1(#1), M2(#1), M3(#1), M0(#2), M1(#2),M2(#2), M3(#2), M0(#3), M1(#3), M2(#3), M3(#3),M0(#4), M1(#4), M2(#4), M3(#4), M0(#5), M1(#5),M2(#5), M3(#5), M0(#6), M1(#6), M2(#6), M3(#6),M0(#7), M1(#7), M2(#7), M3(#7).

2) RR arbitration scheme: M0(#0), M0(#1), M0(#2), M0(#3), M0(#4), M0(#5), M0(#6), M0(#7), M1(#0), M1(#1),M1(#2), M1(#3), M1(#4), M1(#5), M1(#6), M1(#7),M2(#0), M2(#1), M2(#2), M2(#3), M2(#4), M2(#5),M2(#6), M2(#7), M3(#0), M3(#1), M3(#2), M3(#3),M3(#4), M3(#5), M3(#6), M3(#7).

3) RL arbitration scheme: M0(#0), M0(#1), M1(#0), M1(#1), M1(#2), M1(#3), M1(#4), M1(#5), M1(#6), M1(#7),M2(#0), M2(#1), M2(#2), M2(#3), M2(#4), M2(#5),M3(#0), M3(#1), M3(#2), M3(#3), M0(#2), M0(#3),M2(#6), M2(#7), M3(#4), M3(#5), M3(#6), M3(#7),M0(#4), M0(#5), M0(#6), M0(#7).

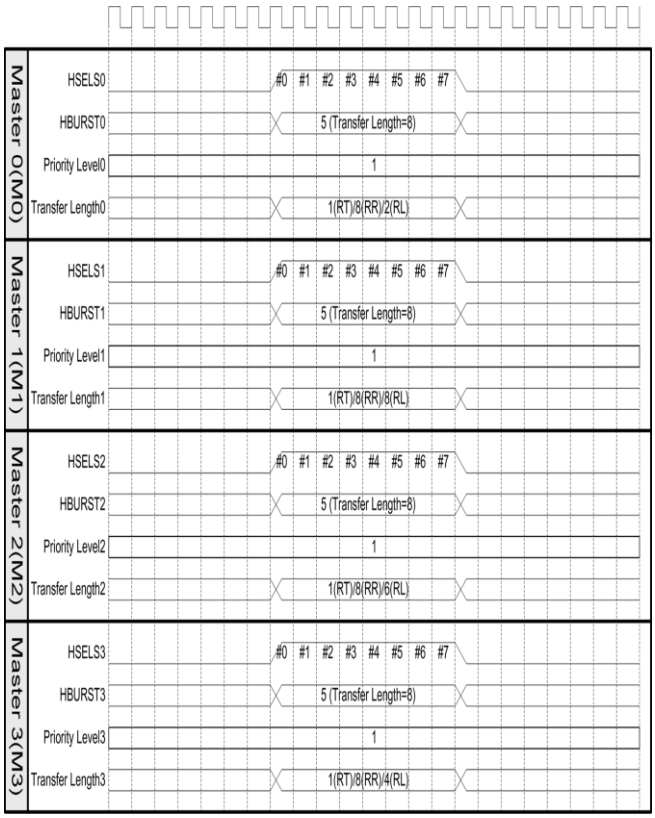

Fig. 2. Configuration for the round robin priority

Dynamic priority scheduling is a type of scheduling algorithm in which the priorities are calculated during the execution of the system. The goal of dynamic priority scheduling is to adapt to dynamically changing progress and form an optimal configuration in self-sustained manner. It can be very hard to produce well-defined policies to achieve the goal depending on the difficulty of a given problem. Earliest deadline first scheduling and least slack time scheduling are examples of Dynamic priority scheduling algorithms.

Fig.3 shows the configurations for the dynamic-priority arbitration schemes. In the dynamic-priority arbitration schemes, the priority of the masters can be changed by the SM demand of each master. Furthermore, the transfer length is assigned as 1 in transfer-based arbitration and 4 in transaction-based arbitration. Also, the transfer length for desired-transfer-length-based arbitration is assigned, the arbitration results are as follows

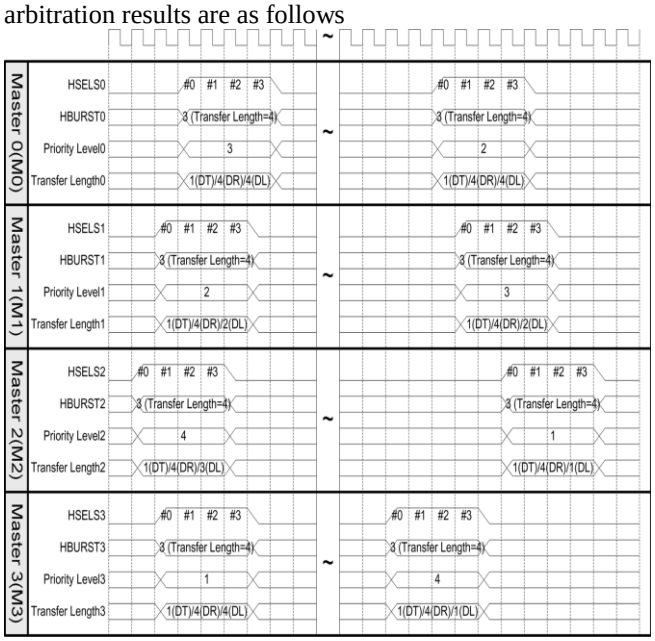

Fig. 3.Configuration for the Dynamic priority

1) DT arbitration scheme: M2(#0), M3(#0), M3(#1),M3(#2), M3(#3), M1(#0), M1(#1), M1(#2), M1(#3),M0(#0), M0(#1), M0(#2), M0(#3), M2(#1), M2(#2),M2(#3) M3(#0), M3(#1), M0(#0), M0(#1), M0(#2),M2(#0), M2(#1), M2(#2), M2(#3), M0(#3),M1(#0),M1(#1),M1(#2),M1(#3),M3(#2),M3(#3).

2) DR arbitration scheme: M2(#0), M2(#1), M2(#2),M2(#3), M3(#0), M3(#1), M3(#2), M3(#3), M1(#0),M1(#1), M1(#2), M1(#3), M0(#0), M0(#1), M0(#2),M0(#3) M3(#0), M3(#1), M3(#2), M3(#3), M0(#0),M0(#1), M0(#2), M0(#3), M2(#0), M2(#1), M2(#2),M2(#3), M1(#0), M1(#1), M1(#2), M1(#3).

3) DL arbitration scheme: M2(#0), M2(#1), M2(#2),M3(#0), M3(#1), M3(#2), M3(#3), M1(#0), M1(#1),M1(#2), M1(#3), M0(#0), M0(#1), M0(#2), M0(#3),M2(#3) M3(#0), M3(#1), M0(#0), M0(#1), M0(#2),M0(#3), M2(#0), M2(#1), M2(#2), M2(#3),M1(#0),M1(#1),M1(#2),M1(#3),M3(#2),M3(#3).

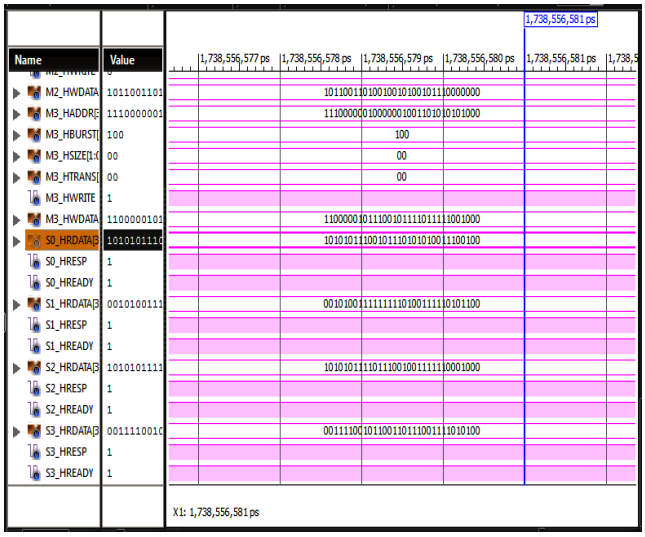

Fig. 4. Simulation results Dynamic arbitration schemes.

Figure 4.Shows the complete simulation results for Dynamic arbitration scheme. We implemented different slave side arbitration schemes for the ML-AHB busmatrix. Each arbitration scheme based busmatrix was implemented with synthesizable RTL VHDL targeting XILINX FPGA (XC2VP100-6ff1704). The XILINX design tool (ISE 7.1i) was used to measure total area. We utilized a ModelSim II simulator to measure the performance of the ML-AHB busmatrix with the different arbitration schemes and demonstrate the efficiency of our flexible SM arbitration.

IV. CONCLUSION

Experimental results show that, although the area of the proposed Dynamic Arbitration scheme is 9%–25% larger than those of other arbitration schemes, proposed arbiter improves the throughput by 14%–62% compared with other schemes. Therefore, it is better to apply this Dynamic arbitration scheme to an application specific system because it is easy to tune the arbitration scheme according to the features of the target system. Multi-layer AHB is an interconnection scheme, based on the AHB protocol that enables parallel access paths between multiple masters and slaves in a system. This is achieved by using a more complex interconnection matrix and gives the benefit of increased overall bus bandwidth, and more flexible system architecture. A key advantage of multi-layer AHB is that standard AHB master and slave modules can be used without the need for modification. In future we are looking to propose this arbiter to AXI protocol.

REFERENCES

1. M. Drinic, D. Kirovski, S. Megerian, and M. Potkonjak, “Latency guided on-chip bus-network design,” IEEE Trans. Comput.-Aided Design Integer, 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 bus matrix,” ETRI J., vol. 28, no. 3, pp. 397–400, Jun. 2006.
3. ARM, “AHB Example AMBA System,” 2001
4. IBM, New York, “32-bit Processor Local Bus Architecture Specification,” 2001.
5. R. Usselmann, “WISHBONE interconnect matrix IP core,” Open-Cores, 2002 [6] N.J. Kim and H.J. Lee, “Design of AMBA wrappers for multiple clock operations,” in Proc. Int. Conf. ICCCAS, Jun. 2004, vol. 2, pp. 1438–1442
6. D. Flynn, “AMBA: Enabling reusable on-chip designs,” IEEE Micro,vol.17,no.4,pp.20–27,Jul./Aug.1997.
7. S. Y. Hwang, H.J. Park, and K.S. Jhang, “Performance analysis of slave-side arbitration schemes for the multi-layer AHB bus matrix,” J. KISS, Comput Syst. Theory, vol. 34, no. 5, pp. 257–266, Jun. 2007.
8. 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 Integer (VLSI) Syst., vol. 15, no. 2, pp. 240–245, Feb. 2007.
9. 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.

AUTHORS PROFILE

Manjunath M. was born in Adakanahalli, Mysore, in 1987. I received the B.E. degree in EEE from VTU, Belgaum, Karnataka, in 2009 and MTECH degree in VLSI Design and Embedded system from VTU, Belgaum in 2012.