IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735. Volume 7, Issue 1 (Jul. - Aug. 2013), PP 31-41
www.iosrjournals.org

Designing of a AMBA-AHB Multilayer Bus matrix Self-Motivated Arbitration scheme

Mr. Ravi. M[1], Mr. Sudhir Dakey[2]
(1)M.E, Department of Electronics & Communication Engineering
(2)Asst.Professor, Department of Electronics & Communication Engineering
(1)(2)MVSREC, Nadargul, Hyderabad.

Abstract: *The AMBA-AHB Multilayer Bus matrix Self-Motivated Arbitration scheme proposed three methods for data transmiting from master to slave for on chip communication. Multilayer advanced high-performance bus (ML-AHB) busmatrix employs 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, in the former, the unit of arbitration can be a transaction or a transfer. However, the ML-AHB busmatrix of ARM offers only transfer-based fixed-pri-ority and round-robin arbitration schemes. In this paper, we propose the design and implementation of a flexible arbiter for the ML-AHB busmatrix to support three priority policies fixed priority, round robin, and dynamic priority and three data multiplexing modes transfer, transaction, and desired transfer length. In total, there are nine possible arbitration schemes. The proposed arbiter, which is self-motivated (SM), 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. Experimental results show that, although the area overhead of the proposed SM arbitration scheme is 9%–25% larger than those of the other arbitration schemes, our arbiter improves the throughput by 14%–62% compared to other schemes.*

I. Introduction

THE ON-CHIP bus plays a key role in the system-on-a-chip (SoC) design by enabling the efficient integration of het-erogeneous system components such as CPUs, DSPs, applica-tion-specific cores, memories, and custom logic [1]. Recently, as the level of design complexity has become higher, SoC de-signs require a system bus with high bandwidth to perform mul-tiple operations in parallel [2]. To solve the bandwidth prob-lems, there have been several types of high-performance on-chip buses proposed, such as the multilayer AHB (ML-AHB) bus-matrix from ARM [3], the PLB crossbar switch from IBM [4], and CONMAX from Silicore [5]. 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 at-tracts many IP designers [6], and the good architecture of the AMBA bus for applying embedded systems with low power [7].

The ML-AHB busmatrix is an interconnection scheme based on the AMBA AHB protocol, which 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 both increased overall bus bandwidth and a more flexible system structure [3]. 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 [8]. The transaction-based arbiter mul-tiplexes 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 usu-ally dependent on the application requirements; recent applica-tions are likewise becoming more complex and diverse. By im-plementing an efficient arbitration scheme, the system perfor-mance can be tuned to better suit applications [9].

For a high-performance on-chip bus, several studies re-lated to the arbitration scheme have been proposed, such as table-lookup-based crossbar arbitration [10], two-level time-division multiplexing (TDM) scheduling [11], token-ring mechanism [12], dynamic bus distribution algorithm [13], and LOTTERYBUS [14].

However, these approaches employ master-side arbitration. Therefore, they can only control pri-ority policy and also present some limitations when handling the transfer-based arbitration scheme since master-side arbitra-tion uses a centralized arbiter. In contrast, it is possible to deal with the transfer-based arbitration scheme as well as the trans-action-based arbitration scheme in slave-side arbitration. In this paper, we propose a flexible arbiter based on the self-motivated (SM) arbitration scheme for the ML-AHB busmatrix. Our SM 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. Fur-thermore, our arbiter provides the desired-transfer-length-based

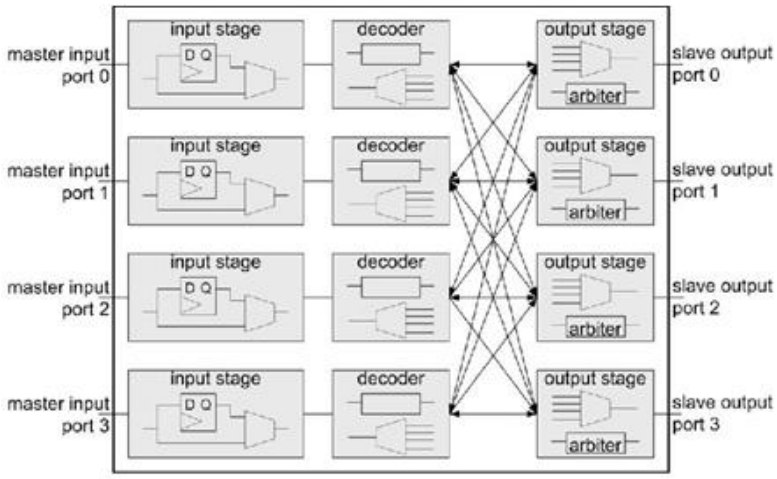

Fig. 1. Overall structure of the ML-AHB busmatrix of ARM [3].

fixed-priority, round-robin, and dynamic-priority arbitration schemes. In addition, the proposed SM 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.

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 SM arbitration scheme for the ML-AHB busmatrix. We then present experimental results in Section IV and concluding re-marks in Section V.

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 [3]. 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 com-mence 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 em-ploys 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, specif-ically 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.

III. Sm Arbitration Scheme For The Ml-Ahb Busmatrix

An assumption is made that the masters can change their pri-ority level and can issue the desired transfer length to the arbiters in order to implement a SM arbitration scheme. This assumption

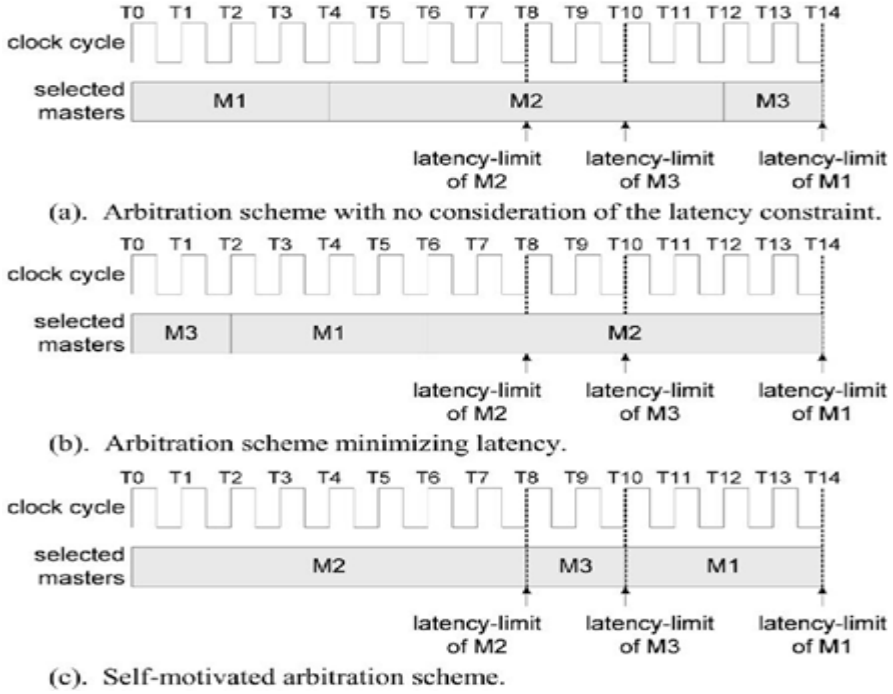

Fig. 2. Arbitration scheme examples in an embedded system. (a) Arbitration scheme with no consideration of the latency constraint. (b) Arbitration scheme minimizing latency. (c) SM arbitration scheme.

should be valid because the system developer generally recog-nizes the features of the target applications [15]. For example, some masters in embedded systems are required to complete their job for given timing constraints, resulting in the satisfac-tion 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. Previous works solved this issue by minimizing the la-tencies 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 [16]. Unlike existing works, our scheme can keep the latency close to its given constraint by adjusting the priority level and transfer length of the masters. Fig. 2 shows an example.

In this example, the service latencies (latency-limit times) of M1, M2, and M3 are 4, 8, and 2 cycles (T14, T8, and T10), re-spectively. The requests for three masters are all initiated at T0, and M3 is the most latency-sensitive master. Fig. 2(a) shows an arbitration scheme that does not use latency constraints for ar-bitration. Therefore, M2 and M3 violate the latency constraint as the masters are selected in ascending order. Only M1 meets the constraint. Fig. 2(b) shows the scheduling of a typical la-tency-minimizing arbiter. It minimizes the latency of the most latency-sensitive module, namely, M3, causing M2 to violate its constraint. Although neither of these two arbitration schemes can meet the latency constraints for all three masters, in the SM arbitration shown in Fig. 2(c), all masters use the bus with no violations by configuring the priority levels (transfer lengths) of M1, M2, and M3 as the lowest, highest, and intermediate prior-ities (4, 8, and 2), respectively.

We use part of a 32-b address bus of the masters to inform the arbiters of the priority level and the desired transfer length

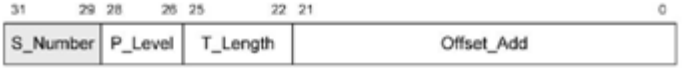

Fig. 3. Decoding information of the 32-b address bus.

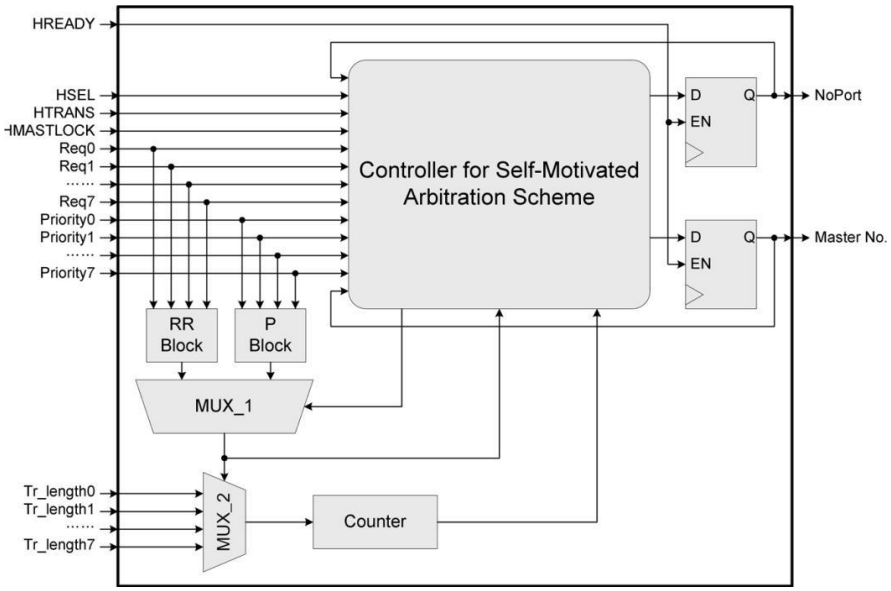

Fig. 4. Internal structure of our arbiter.

of the masters. Fig. 3 shows the decoding information for our address bus.

In Fig. 3, *S_Number* indicates the target slave number, *P_Level* means the priority level of a master, *T_Length* denotes the desired transfer length of a master, and *Offset_Add* specifies the internal address of the target slave. Each of *S_Number* and *P_Level* consists of 3 b because the maximum number of master–slave sets is 8 ×8 [3]. Also, *T_Length* is composed of 4 b because the maximum number of burst lengths is 16 [3]. Although we used 7 b for *P_Level* and *T_Length* in the 32-b address bus to notify the arbiters of the priority level and the desired transfer length of a master, we consider it adequate to express the internal address of a slave because the range of *Offset_Add* is from 0 to $2^{22}-1$. Through the aforementioned assumption, the priority level and transfer length can then be changed by the SM demand of each master.

Fig. 4 shows the internal structure of our arbiter based upon the SM arbitration scheme.

In Fig. 4, the *NoPort* signal means that none of the mas-ters 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 gen-erated by the controller for the SM arbitration scheme. In gen-eral, our arbiter consists of an RR block, a P block, two multi-plexers, 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 at-tempts 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. 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. 5. We then generate the up- or down-masked vector created through bitwise AND-ing operation between the mask vector

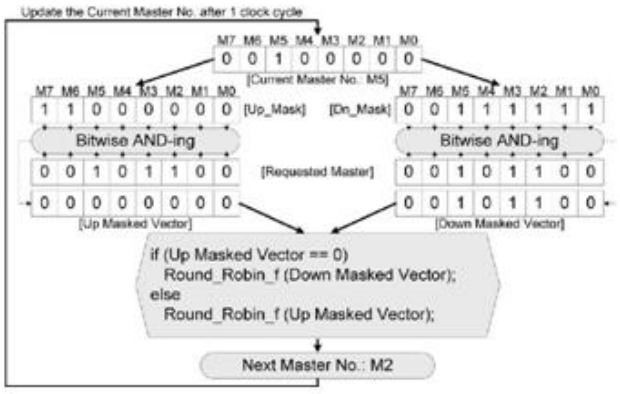

Fig. 5. Internal process of the RR block.

```
1: function Round_Robin_f (Masked_Vector: std_logic_vector)
2: return integer is
3: variable Next_Master_No : integer;
4: begin
5:     Next_Master_No := 0;
6:     for i in Masked_Vector'left downto 0 loop
7:         if (Masked_Vector (i) = '1') then
8:             Next_Master_No := i;
9:         end if;
10:    end loop;
11:    return Next_Master_No;
12: end;
```

Fig. 6. VHDL code of the round-robin function.

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. 5.

Fig. 6 shows the VHDL code of the round-robin function at the behavioral level.

In Fig. 6, a master for the next transfer is selected through the for-statements in line 6, with the priority level of the least significant bit in *Masked_Vector* being the highest. If we modify the range of *Masked_Vector* in line 6 to "*0 to Masked_Vector'left*," then the priority level of the most signifi-cant bit in *Masked_Vector* becomes the highest.

Fig. 7 shows the internal procedure of the P block. First of all, we create the highest priority vector (V) through the round-robin function of Fig. 6. 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 func-tion, while the current master is updated after 1 clock cycle.

Fig. 8 shows the VHDL code of the priority function at the behavioral level.

In Fig. 8, the master with the highest priority is selected through the for-statements in line 7.

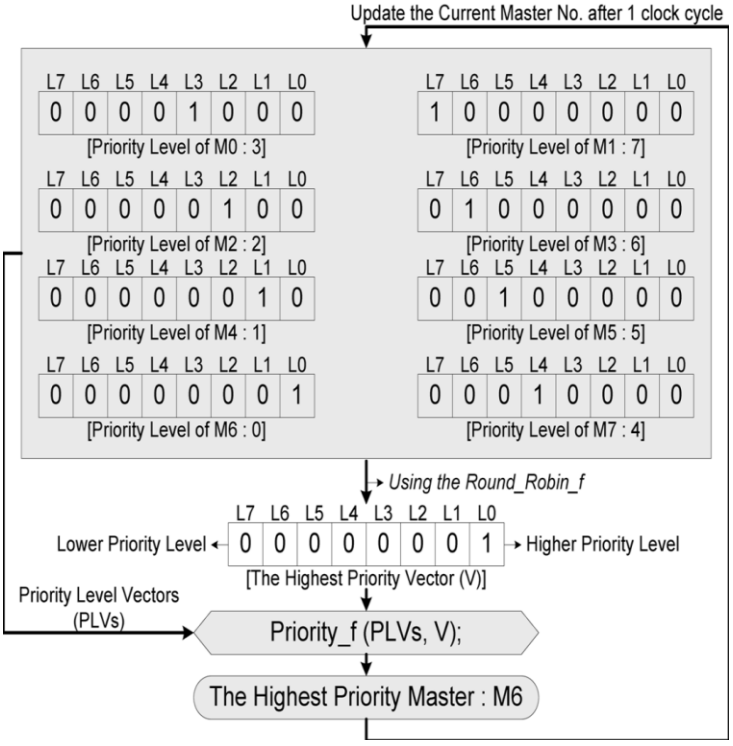

Fig7: internal procedure for **P** block

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.

The SM 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-priorityarbitration 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 ransfer based upon a single transfer (burst transaction), and the desired-transfer-length-based arbiter multiplexes the datatransfer based on the transfer length assigned by the masters.

Fig. 9 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, he 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 (for example, let , , , and). The arbitration results of Fig. 8 are as follows ("#" indicates the transfer number).

1) FT arbitration scheme: 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), M(#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).

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).

In this case, the result of transaction-based arbitration is equalto that of desired-transfer-length-based arbitration because the priority levels of all the masters are fixed.

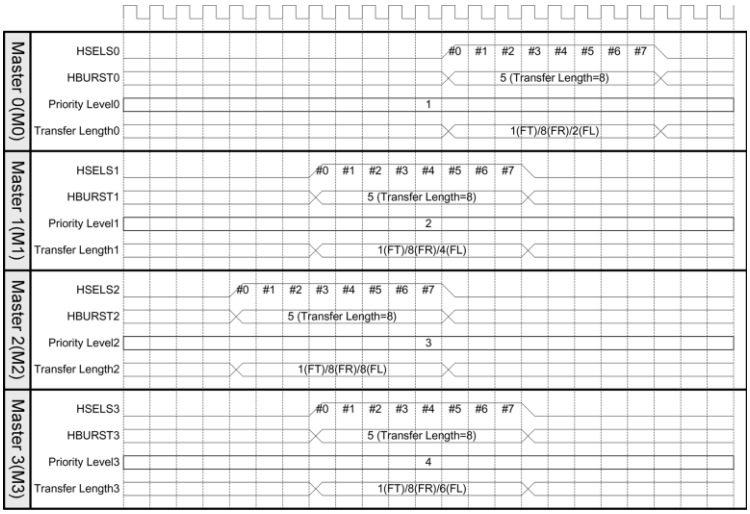

Fig8:fixed priority arbitration scheme

Fig. 9 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-transferlength based arbitration, the transfer length is assigned by the demand of each master (for example, let , , , and). The arbitration results of Fig. 10 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), 0(#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. 10 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, as shown in Fig. 10. The arbitration results of Fig. 10 are as follows.

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).

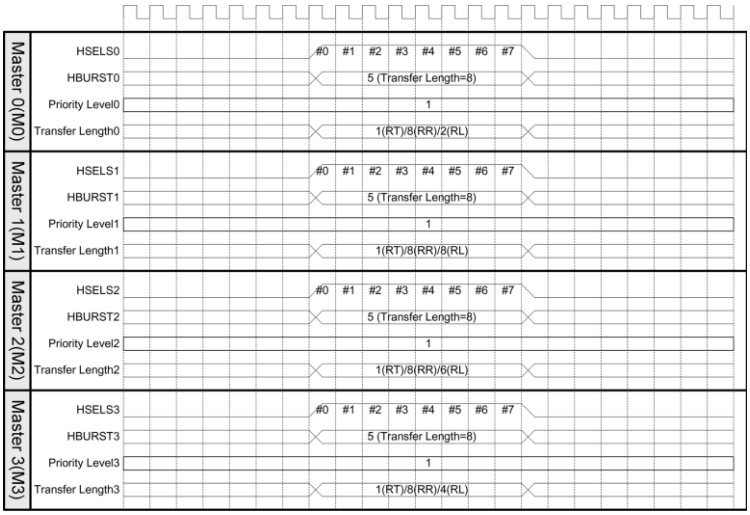

Fig9:round robin arbitration scheme

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).

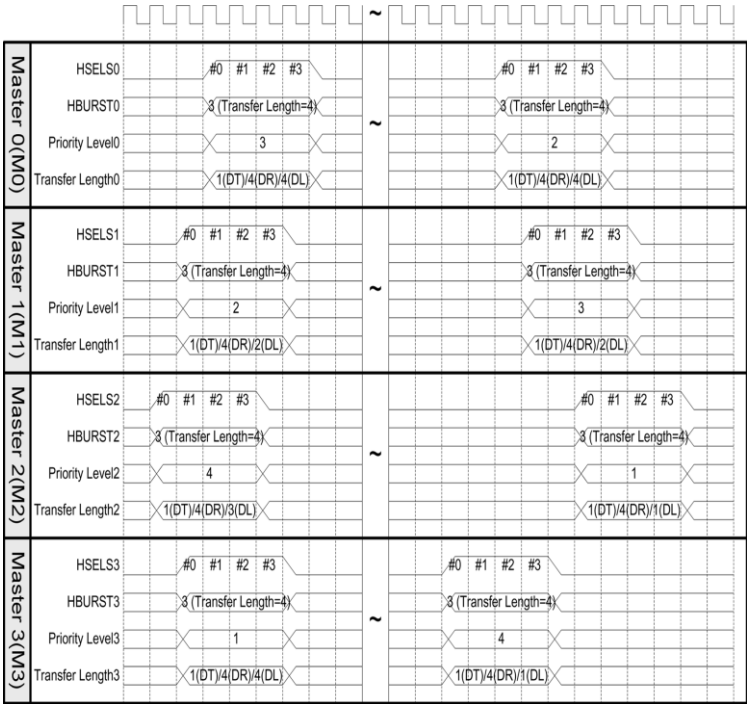

Fig10:Dynamic arbitration scheme

IV. Designing Results of Ahb

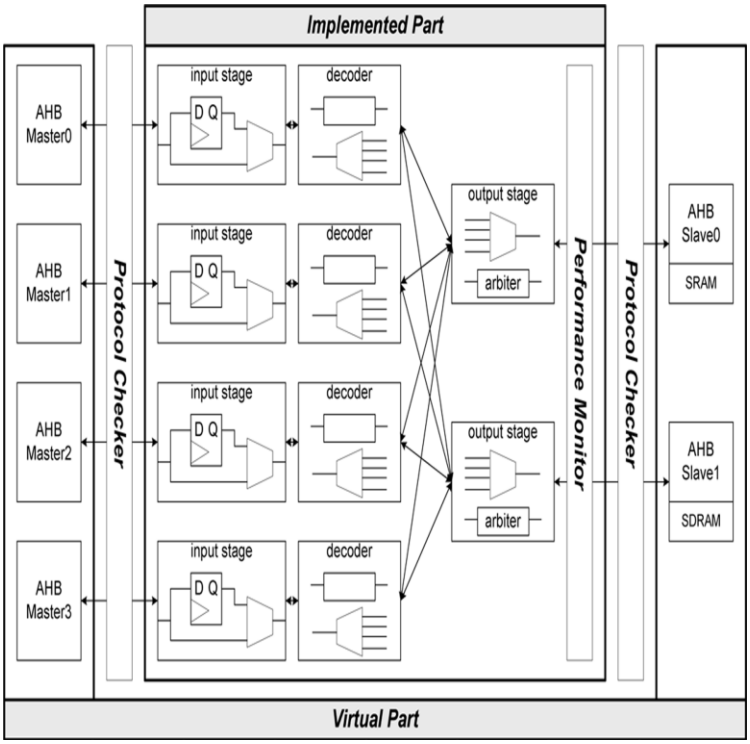

Fig11:Simulation environment for performance analysis.

ML-AHB Busmatrix desind by using of fig11 architecture , using of hdl language . simulated done by using of modelsim6.4b tool, synthesis by Xilinx ISE 10.1 tool, these results are shown in below figure 12&13.

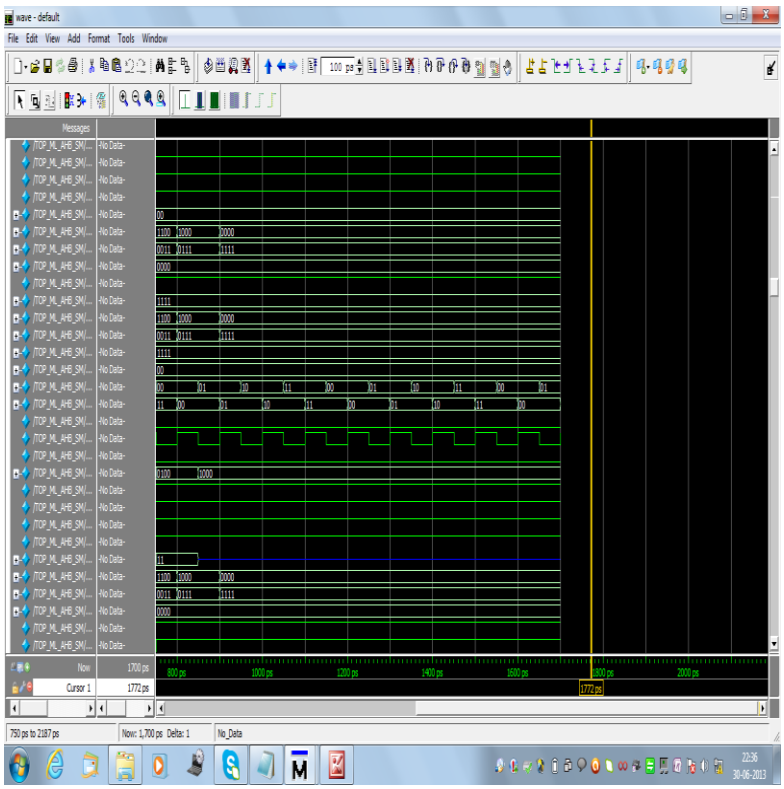

Fig12: ML-AHB simultion wave form.

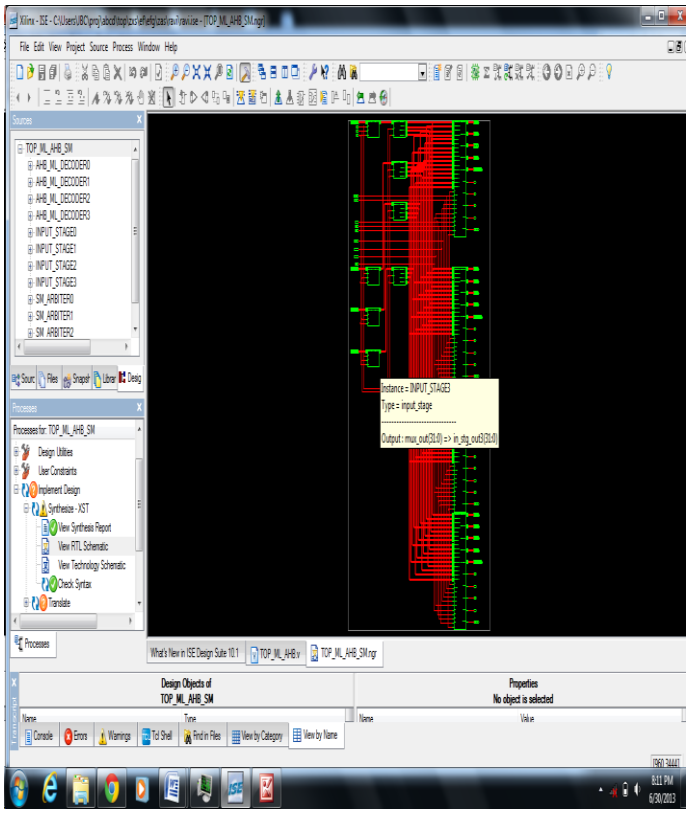

Fig13: ML-AHB synthasesis RTL view.

V. Conclusion

In this paper, the 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 SM 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.

This design proposed ML- AHB SM arbitration schemes increases area than the other arbitration schemes in ML-AHB, but ML-AHB SM arbitration scheme gives the better performance when it selects the input stage and output stage in self motivated manner. Therefore expect that it would be better to apply our SM 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 multipleclock operations," in *Proc. Int. Conf. ICCCAS*, 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.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] J. H. Han, M. Y. Lee, B. Younghwan, and C. Hanjin, "Application specific processor design for H.264 decoder with a configurable embedded processor," *ETRI J.*, vol. 27, no. 5, pp. 491–496, Oct. 2005.
[16] M. Jun, K. Bang, H.-J. Lee, N. Chang, and E.-Y. Chung, "Slack-based bus arbitration scheme for soft real-time constrained embedded systems," in *Proc. Int. Conf. ASP-DAC*, Jan. 2007, pp. 159–164.
[17] 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.
[18] E.-G. Jeong, J.-G. Lee, K.-S. Jhang, J.-A. Lee, and D. Har, "Asynchronous layered interface of multimedia socs for multiple outstanding transactions," *J. VLSI Signal Process. Syst.*, vol. 46, no. 2/3, pp. 133–151, Mar. 2007.
[19] S. Y. Hwang, H. J. Park, and K. S. Jhang, "An implementation and performance analysis of slave-side arbitration schemes for the ML-AHB busmatrix," in *Proc. Int. Conf.*