International Journal For Technological Research In Engineering
Volume 3, Issue 10, June-2016 *ISSN (Online): 2347 - 4718*

IMPLEMENTATION OF AMBA BUS ARBITER USING ROUND ROBIN SCHEME

Ragini Soni[1], Pravin Tiwari[2]
[1]M. Tech. Student, [2]Asst. Professor
Department of Electronics and Communication
Takshshila Institute of Engineering and Technology, Jabalpur, India

ABSTRACT: In System on Chip (SoC) buses, intellectual properties (IPs) need to communicate with each other to access the required functionality. When the SoC bus is connected with more IPs, contentions occur while multiple IPs requests the bus at the same time. This makes on-chip bus based communication a major challenge for the system designer in the current SoC technology. The communication architectures must be able to adapt themselves according to the real-time requirements of the IPs. Hence, bus arbiters are proposed. The arbiter block plays important role in the SoC shared bus communication. The masters on a SoC bus may issue requests simultaneously and hence an arbiter is required to decide which master is granted for bus access. Bus Arbiter plays a vital role in handling the requests from the master and responses from slave (like Acknowledgement signal, Retry, etc). The main objective of arbitration algorithms is to ensure that only one master has access to the bus at any given time, all the other masters are forced to remain in the idle state until they are granted the use of the bus.
KEYWORDS: SoC, On-chip bus, dynamically configurable arbiter, latency

I. INTRODUCTION

As the era of a billion transistors on a single chip fast approaches, more Processing Elements (PEs) can be placed on a System-on-a-Chip (SoC). Most PEs in an SoC communicate with each other via buses and memory. As the number of bus masters increases in a single chip, the importance of fast and powerful commands are necessary. This makes on-chip bus based communication a major challenge for the system designer in the current SoC technology. The communication architectures must be able to adapt themselves according to the real-time requirements of the PEs. Hence, bus arbiters are proposed. The arbiter is a electronic devices that allocate access to shared resources. Arbiter block plays important role in the SoC shared bus communication. The masters on a SoC bus may issue requests simultaneously and hence an arbiter is required to decide which master is granted for bus access. Bus Arbiter plays a vital role in handling the requests from the master and responses from slave (like Acknowledgement signal, Retry, etc). The main objective of arbitration algorithms is to ensure that only one master has access to the bus at any given time, all the other masters are forced to remain in the idle state until they are granted the use of the bus.

The arbiter has 2 schemes as follows.

- Round Robin scheme
- Fixed priority scheme

A particular scheme can be programmed as required. The round-robin scheme is about time-slicing that is we must fix a certain amount of time when each process must be executed. It is usually implemented using equal priority for simplicity. If the tasks have a relatively equal importance, then the round-robin works better, since all the tasks get a better chance of getting run; we avoid the situation where the task with the lowest priority hardly ever gets run, since there seems to always be another task with a higher priority. Imagine we need to read data from a number of sources. Basically, they are all important, so we would probably choose this scheme. In fixed priority scheme, every master is programmed with its own priority.

II. BUS ARBITER

The arbiter block plays important role in the SoC shared bus communication. The masters on a SoC bus may issue requests simultaneously and hence an arbiter is required to decide which master is granted for bus access. In many applications, masters may have real-time and/or bandwidth requirements on requests. A master with a real-time requirement demands its transactions accomplished within a fixed number of clock cycles. On the other hand, a master with a bandwidth requirement must occupy a fixed fraction of total bandwidth of a bus. The arbitration algorithm could be implemented in a centralized or a distributed fashion. In a centralized arbitration scheme, the master side of the arbitration protocol instantiated in each master communicates with the slave side of the arbitration protocol instantiated in an additional arbiter component attached to the bus. In a distributed arbitration scheme, there is no slave side of the arbitration protocol and the master sides of the protocol in each master regulate accesses among themselves. Bus Arbiter plays a vital role in handling the requests from the master and responses from slave (like Acknowledgement signal, Retry, etc). The available arbitration protocols strive to optimize the contentions arising while different masters issue the request for using the bus at the same time. The arbitration algorithms must also be in an optimized manner to handle the contingencies. The main objective of arbitration algorithms is to ensure that only one master has access to the bus at any given time, all the other masters are forced to remain in the idle state until they are granted the use of the bus. The power utilized by the arbitration

technique in different on-chip communication tends to vary significantly for a particular application. Thus it makes the comparison of arbitration algorithms a vital step in the SoC design.

III. BLOCK DIAGRAM OF BUS ARBITER

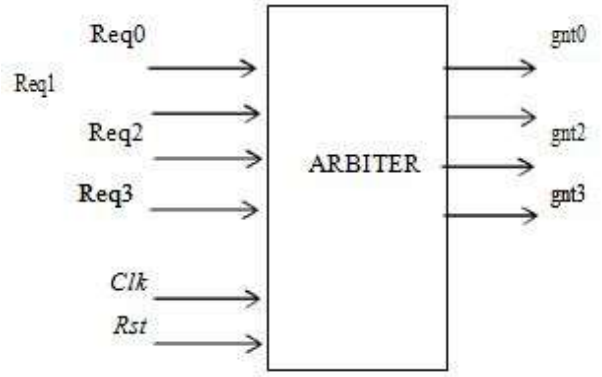

Fig. 3.1 Basic block diagram of bus arbiter

The above Fig.3.1 shows the basic block diagram of bus arbiter. Here for simplicity we are considering only four requests. The inputs to the bus arbiter are

Req0 - request signal generated from processor 1 Req1 - request signal generated from processor 2 Req2 - request signal generated from processor 3 Req3 - request signal generated from processor 4 Clk – clock signal

Rst – reset signal

The outputs of the arbiter are

Gnt0 – grant signal for processor 1 in order to acquire cpu& perform data transfer Gnt1– grant signal for processor 2 in order to acquire cpu& perform data transfer

Gnt2 – grant signal for processor 3 in order to acquire cpu& perform data transfer

Gnt3 – grant signal for processor 4 in order to acquire cpu& perform data transfer

IV. LOGIC DIAGRAM OF 4X4 BUS ARBITER BLOCK

A round-robin token passing bus or arbiter guarantees fairness (no starvation) among masters and allows any unused timeslot to be allocated to a master whose round-robin turn is later but who is ready now. A reliable prediction of the worst-case wait timeis another advantage of the round-robin protocol. The worst-case wait time is proportional to number of requestors minus one. The protocol of a round-robin token passing bus or switch arbiter works as follows. In each cycle, one of the masters (in round-robin order) has the highest priority (i.e., owns the token) for access to a shared resource. If the token-holding master does not need the resource in this cycle, the master with the next highest priority who sends a request can be granted the resource, and the highest priority master then passes the token to the next master in round-robin order. Here a BA is generated to handle four requests. Figure 4 shows the BA block diagram for four bus masters. To generate a BA, Round robin arbiter generator(RAG) takes as input the number of masters and produces synthesizable Verilog code at the RTL level. The generated BA consists of a D flip-flop, priority logic blocks, an M-bit ring counter and M-input OR gates as shown in Figure 4 where M=4. A

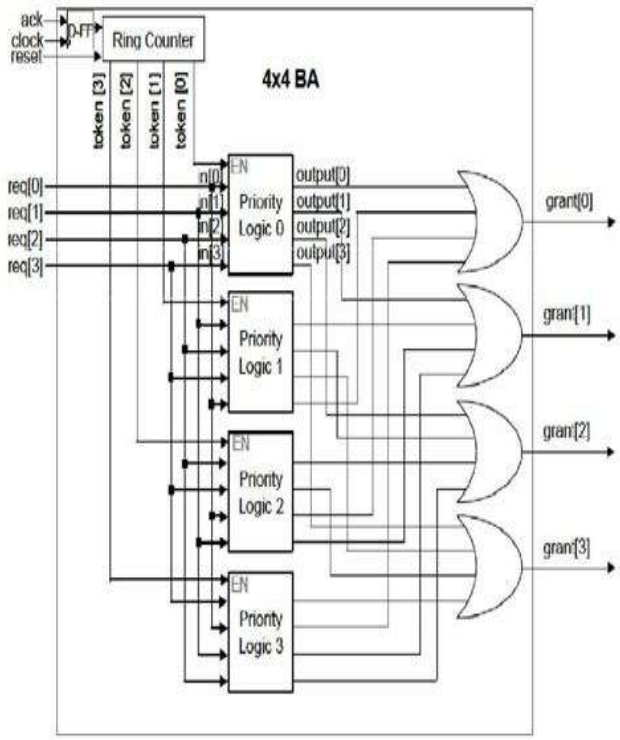

Fig 4.1 Logic Diagram of 4x4 Bus Arbiter

The priority of inputs are placed in descending order from in[0] to in[3] in the priority logic blocks (Priority Logic 0 through 3) shown in Figure 5 Thus, in[0] has the highest priority, in[1] has the next priority, and so on. To implement a BA, we employ the token concept from a token ring in a network. The possession of the token allows a priority logic block to be enabled. Since each priority logic block has a different order of inputs (request signals), the priority of request signals varies with the chosen priority logic block. The token is implemented in a 4-bit ring counter as shown in Figure 5. The outputs (four bits) of the ring counter act as the enable signals to the priority logic blocks. Thus, only one enabled priority logic block can assert a grant signal. The ack signal to the bus arbiter is delayed by one arbitration cycle by a D flip-flop as shown in Figure 5. The delayed ack signal pulls a trigger to the ring counter so that the content of the ring counter is rotated one bit. Thus, the token bit is rotated left each cycle, with 4'b1000 rotating to 4'b0001 in Figure 5 and the token is initialized to one at the reset phase (e.g., 4'b0001 for four-bit ring counter) so that there is only one '1' output by the ring counter. In the round-robin algorithm, each master must wait no longer than (M-1) time slots, the period of time allocated to the chosen master, until the next time it receives the token (i.e., highest priority). The assigned time slot can also be yielded to another master if the owner of the time slot has nothing to send. This protocol guarantees a dynamic priority assignment to bus masters (requestors) without starvation.

V. STATE DIAGRAM FOR BUS ARBITER

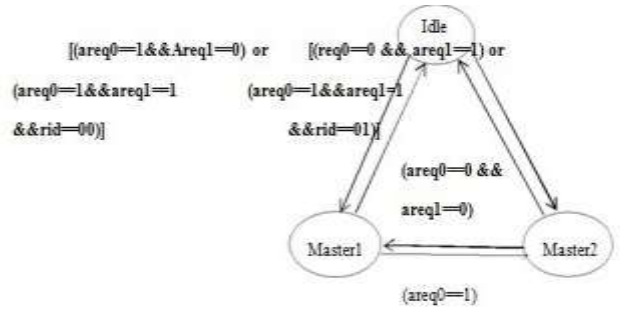

Fig.5.1 State Diagram for Bus Arbiter

State diagram model is used for modeling reactive or event driven embedded systems whose processing behavior are dependent on state transitions. The model describes the system behavior with 'states', 'events', 'actions' & 'transition'. State is a representation of a current situation. An event is an input to the state. The event acts as stimuli for state transition. Transition is the movement from one state to another. Action is an activity to be performed by the state machine. Here we are considering only two masters are requesting for bus access. It consists of three states-

- Idle
- Master1
- Master2

areq0 & areq1 are the requests generated from master1 & master2 respectively. If master1 requests for cpu access then areq0 is set to one & areq1 signal is forced to be in idle state. State transition of master1 takes place from idle to master1 state. After the completion the data transfer if in case areq1 is set the transition takes from master1 to master2. If areq1 is not set then transition from master1 to idle state occurs. If master2 requests for cpu access then areq1 is set to one & areq0 signal is forced to be in idle state. State transition of master2 takes place from idle to master2 state. After the completion the data transfer if in case areq0 is set the transition takes from master2 to master1. If areq0 is not set then transition from master2 to idle state occurs.If both the master's requests are set i.e,areq0=1 && areq1=1 then it depends on the internal signal rid. If rid signal is set 00 then master1 gets the access. If rid value is set to 01 then master2 gets the access.

VI. PROPOSED ARBITER DESIGNS

The proposed arbiter designs namely Real Time_Staticpriority(RT_SP), realTime_Roundrobin. The proposed rbiter designs namely, real Time_staticpriority (RT_SP),Real_Time_Staticpriority(RT_SP)RealTime_Round robin(RT_RB), Two-level Dynamic Scheduler and Three-level Dynamic Scheduler are discussed in this section.

A. RealTime_Staticpriority (RT_SP)

The main aim of this proposed RT_SP is to reduce the starvation problem present in the conventional Static Priority algorithm at heavy traffic conditions. In the proposed RT_SP, three methods have been modeled to ensure low starvation at heavy traffic conditions.

A.1. RT_SP Method I

In this method , following steps are done:

1. The masters are classified according to their priority.
2. The common warning line value is initialized.
3. The requests from the masters are issued.
4. Static priority algorithm grants the highest priority master.
5. The warning line value is checked. If the limit is not reached then the highest priority master continues its data transfer. But if the limit is reached, comparison of the remaining pending requests from the masters is done.
6. Now, the highest priority request out of the pending requests is granted.
7. Again, Static priority based grant system is followed
8. Steps 4 and 5 are repeated till no request is issued from any master. When there is no request to be processed Arbitration process stops.

Common warning line: - The purpose of this warning line is to consider the lower priority masters which are in waiting state, to get grant after a particular number of requests issued. Initially a value is set for common warning line. This value will decrement from the assigned value each time a 'grant' gets generated by the static priority algorithm.

A.2. RT_SP Method II

In Method II and Method III the steps are similar as in Method I and only the way of setting the warningline varies. In this Method II, each master has a separate warning line. So, when a master has reached its warning line limit it will be immediately granted the bus, after the current transaction is completed. Each master can assign a different value to its warning line.

A.3. RT_SP Method III

In this method each master has a separate warning line. Whenever a master issues request, its warning line is set to wait for particular time duration untilthat master gets grant signal. Once the waiting timeexceeds the limit, that particular master gets grantedimmediately after the current transaction getscompleted.

B. Real time _ Round Robin (RT_RB)

The main aim of this proposed algorithm is to eliminate the inefficiency of existing round robin\ algorithm. The disadvantage of the existing round robin arbiter is, it issues grant even if a master has not sent any request. As a result, that bus cycle will be wasted. But this proposed model will issue grant to only requested masters in a round robin fashion. The RT_RB arbiter block diagram is shown in Fig.5.1.The arbiter works in the following manner:

- The masters issue their requests.
- The masters' requests will enable the first OR gate and the OR gate issues a 'HIGH' output signal.
- That output signal will be and operated with request signal of master 1 (Req1).
- If the request from Master1 is present, i.e. Req1 is HIGH, and then grant is issued to Master1.
- Until the Ack signal is received from the slave, grant signal to next master, Master 2 is suspended.

Immediately after the arrival of Ack signal it is and operated with Req2. If both inputs are HIGH, then the Andgate issues grant signal to Master2. After Master2 has completed its transfer, grant is issued to Master3. After Master3 has completed its transfer, grant is issued to Master4. Thus the arbiter grants thebus to the masters in a round robin fashion.

- If the request is not present for a master, then grant signal is given to the next master in the order (through the inverter at the output of the Andgate).

The advantages of RT_RB are as follows:

- Simple structure
- Latency between consecutive *grant* signals is Reduced Area and power dissipation is low.

Removes the disadvantage of existing round robin arbiter

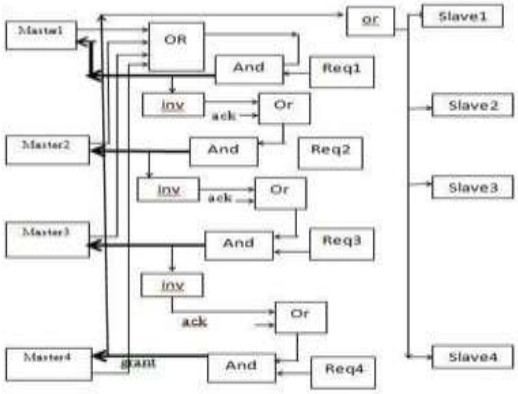

Fig.6.1. RT_RB

VII. SIMULATION RESULTS

The simulation of round robin bus arbiter is done using Xilinx (software). The figures below shows the screenshot of simulated result of above code is shown in below figures. The master bus sends a request to arbiter to access the bus, if the bus is free to access then it sends a grant signal back to the master. Thus the requesting and the grant signal will be high and the rest will be in the idle state(among four master bus three will be in idle state).

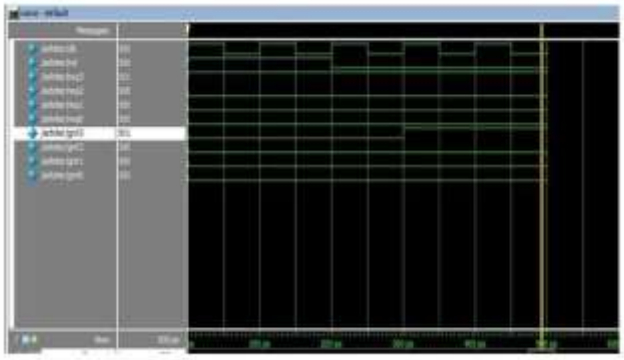

Fig.7.1 simulation result

VIII. CONCLUSION

In this we have discussed the design and the pre-layout simulation using verilog of a Round robin bus arbiter. The arbiter provides very simple system architecture and the latency of the wire delay is very small. In a multiprocessor System-on-a-Chip (SoC) environment, a silicon CMOS chip designer should consider the need for an arbiter to resolve conflicts on shared resources (i.e., bus or equivalent communication channels) among multiple bus masters (e.g, processors). In a bus-based system, processors could be stalled because of bus conflicts. Thus, a high-performance arbiter is needed to resolve bus contentions among bus masters; such a fast arbiter can also reduce processor stall time by shortening arbitration delays. Since the drawback of round robin arbiter is that the critical data transfers may have to wait for a long time before they can proceed, it is possible to combine two arbitration schemes to create a two level arbitration scheme that is two level TDMA/RR arbitration scheme. It is a combination of advantages of two schemes. In this scheme, TDMA arbitration scheme allocates time slots to various masters. If a master does not have any data to transfer during its time slot, a second level Round Robin scheme selects another master to grant bus access to. Such a scheme thus enables better utilization of the bus, compared to the TDMA and RR scheme.

REFERENCES

[1] E. S. Shin, V. J. Mooney III, G. F. Riley, "Round-robin Arbiter Design and Generation," Georgia Institute of Technology, Atlanta, GA, Technical Report GIT-CC-02-38, 2002, Available HTTP: http://www.cc.gatech.edu/tech_reports

[2] "Round-robin arbiter design and generation," in Proceedings of the International Symposium on System Synthesis, pp. 243–248, October 2002.

[3] A paper on "Design and Analysis of Dynamically Configurable Bus Arbiters for SoCs" byS.HemaChitra, P.T.Vanathi.

[4] "SYSTEM-ON-A-CHIP VERIFICATION-Methodology and Techniques" by PrakashRashinkar, Peter Paterson, Leena Singh, Kluwer Academic Publishers.

[5] Alex A. Aravind, "An Arbitration algorithm for multiport memory systems", IEICE Electronic Express, Vol. No2, No.19, 488-494, Oct 2005.

[6] Bu-chung Lin, Geeng-Wei Lee, Juninn Dar Huang and Jing-Yang Jou, "A Precise bandwidth Control Arbitration Algorithm for Hard Real-Time SOC Buses", DAC 2007, pages 165-170.

[7] KanishkaLahiri and AnandRaghunathan, "Lotterybus: A new high-performance communication architecture for System-on-chip Designs", DAC 2001, June 18-22, 2001, ACM, USA.

[8] Massimo Conti, Marco Caldari, Giovanni B.Vece, Simone Orcioni, Claudio Turchetti, "Performance Analysis of different Arbitration Algorithm of the AMBA AHB Bus," DesignAutomation Conference,(DAC'04),2004, 41stVolume , Issue , 2004 Page(s): 618 - 621

