International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Communication between Multiple Resources Using Arbiter Design

Shital S. Horte, Dr. D. V. Padole

[1, 2]G. H. Raisoni College of Engineering, Nagpur, India

Abstract: ***This project describes a important circuit which is called an arbiter to be used in a big designs called switch to communicate in between multiple resources or users. The design description gives results according to suggested implementation for the circuit. And this structure i.e. Arbiter calculates the overall performance of the system-on-chip design. At the end, possibilities for addition, revision and testing structure for an integrated circuit implementation of the arbiter will be considered. The main contribution of this paper is the design and optimization of arbiter circuit. When Circuits require to be constructed out of several self-timed parts, the arbitration is frequently required for the asynchronous design. This paper will give design ideas for operatively interfacing to an arbiter and carry out some research for coding styles for some common arbitration schemes. We consider here the designing of the general purpose arbiter using M resources to N clients. Here in this paper we are going to see static as well as dynamic arbitration techniques.***

Keywords: DSP-Digital Signal Processor, SoC-System on chip, Shared Bus, TDM-Time division Multiplexing

1. Introduction

The arbiters are a vital piece of the scheduler design in which Grant and Request signals are identically designed. A multi-client shared bus system uses a arbiter to give decision or conclusion that which bus client will be getting access to control the shared bus for each cycle of bus. There are many systems exist in which a large number of requesters must approach a common resource. The common resources may be a shared memory, a state machine, a networking fabric switch, or a complex computational element. An arbiter is needed to share the resources or clients among the many requesters. When considering an arbiter into a design, many factors must be considered.

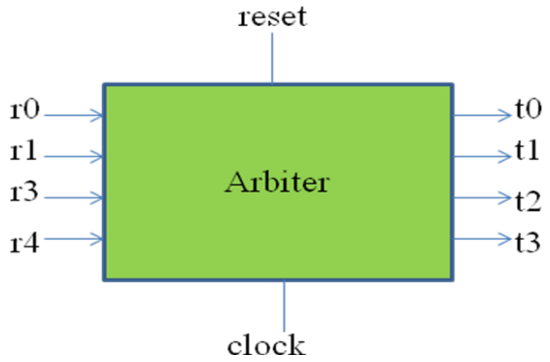

Figure 1: Block Diagram for Synchronous arbiter

The interaction between the users and the arbiter must be proper and suitable for the size and speed of the arbiter. Even, the coding style used will usually have impact on the synthesis results. Arbiter works on three processes request, grant and Accept.

1.1 Process 1 Request Signal

Each input user sends a request to every or requred output resource.

1.2 Process 2 Grant Signal

In an unmatched output receives any requests, it chooses the one which appears next in a fixed, round-robin scheduling technique starting from the highest priority resource. The output denotes each input whether its request was granted or not. The pointer to the highest Priority resource of the round-robin scheduling is incremented (modulo N) to the one location Beyond the granted input if the grant is accepted in Process 3 of the first iteration.

1.3 Process 3 Accept Signal

If an unmatched input user receives a grant, it accepts the one that appears next in a fixed scheduling and round-robin scheduling starting from the highest priority resource.

2. Arbiters can be design in three ways as follows

2.1 Arbiter design with one client many resources

Here in this one user will send request for many resources to have access over it.

2.2 Arbiter design with many client one resources

Here in this type many clients will send request for only one resource depending on used algorithm or technique one particular client will get access to that single resource.

2.3 Arbiter design with many client many resources

Arbiter design with many clients many resources: Here in such type of Arbiter design many clients will send request to many resources and according to used scheduling techniques and algorithms the client get grants to various resources. And these scheduling techniques may be static or dynamic techniques.

3. Considerations for Arbiter Design

There are three independent request signals say r1, r2, r3. In the project four inputs are given as in detailed when the input is given through the signals like r1, r2, r3, r4 and after the processing of these signals it shows results in the form of g1, g2, g3,g4 which are nothing but grant signals. It must be assumed that the priority of the request signals in the order r1>r2> r3>r4.In this project the highest priority among the request signals is r1, then r2, then r3 and lowest priority is r4. Here in this project highest priority workload given to the highest priority input called r1, then according to work it is distributed with another signals. When the high priority workload is given to the high priority input then when request will send to the high priority input then acknowledgement will also send immediately and process on that particular work get starts on the input. Now here we have to consider this situation also if input is given to r2 before r1 then first r1 input get access then after completing the task then it will move to r2 signal i.e. in simple words it will respond after the r1 execution.

For this duration of access time (i.e. Timeout period) is programmed through the various independent processors and the data bus. Duration of execution time to the request (time out period) plays a very important role for executing the input request signals. If any signal is given to the input according to the fixed priority but arbiter does not respond in given time then this whole process repeat again in certain time period and same acknowledgement signal is given by certain time and grant signals get generate in that particular timeout period then process further starts, otherwise further process become stop and process of acknowledge and grant repeat and again arbiter at any instant of time will be in one of the following states:

- g1
- g2
- g3
- g4
- idle

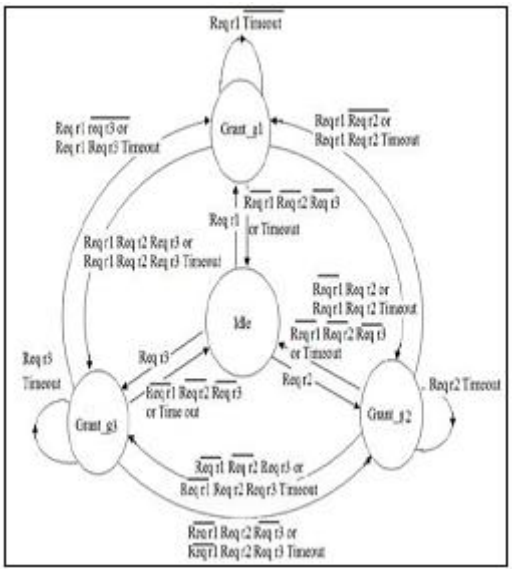

Figure 2: State diagram of synchronous arbiter

4. Methodology

Arbitration Schemes

The asynchronous arbiter plays an important role in the SoC shared bus communication. The clients on a SoC bus may requests simultaneously for the same resource and hence an arbiter is required to decide which client is granted for bus access. And for this an arbiter requires arbitration techniques which are as follows:

4.1 Static Lottery bus Arbitration Scheme

The core of the LOTTERYBUS arbitration scheme is a probabilistic arbitration algorithm implemented in a lottery manager for each bus in the SoC communication architecture. This architecture does not take for granted any fixed communication topology.

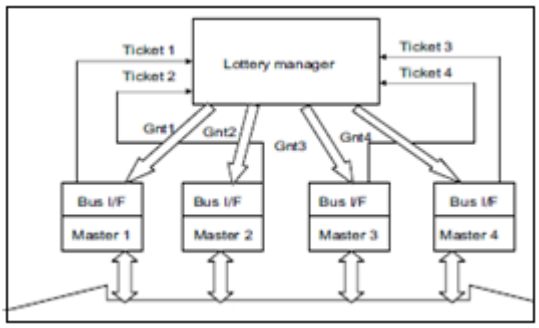

Figure 3: Lottery manager for shared bus

Hence, the various SoC components must be interconnected by a flat, system wide bus or an arbitrary network of shared channels. The lottery manager gathers requests for the access of the shared bus from one or more clients or masters , which are (statically or dynamically) allocated to a number of "lottery tickets" like in above figure 3.This manager which is shown in above figure pseudo-randomly chooses one of the engaging clients to be the winner of the lottery, favoring clients that will have a larger number of tickets, and allows access to the chosen client for some number of shared bus cycle. However, to prevent a client from holding or obtaining the shared bus, a max transfer size is used to limit the number of bus cycles for which the granted client can utilize the bus. The inputs to the lottery manager are a set of requests (one per each client) and the number of tickets held by each master. The output is a set of grant signals (again one per each client) that indicate which client is allowed to transfer data across the bus. The arbitration decision for shared bus is based on a lottery. If there is only one request, a lottery results in granting the bus to the that requesting client. [10,13]

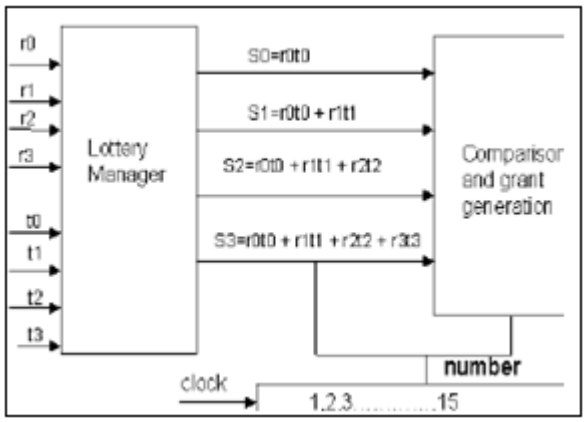

Figure 4 : Block Diagram for Lottery Bus Architecture

Figure 4 shows block diagram of Lottery Bus architecture. And it is having below mentioned three basic blocks.

- Lottery Manager
- Comparator
- Pseudo Random Number Generator

4.2 Dynamic Lottery bus Communication Architecture

In this architecture (figure 5), the inputs to the lottery manager consist of request lines (r0r1r2r3), and the number of tickets which are possessed by each requesting client that are created i.e. generated by ticket generator. Ticket generator generates the ticket like t0, t1, t2, t3. If ticket lines t0,t1,t2 and t3 are 1,2,3,and 4 then in the immediate clock cycle, tickets of clients 0,1,2,3 generated by ticket generator are 2,3,4, and 5. Therefore at each and every lottery, the lottery manager requires to calculate for each resource element Ci, the partial sum $\sum r_j * t_j$. This must be implemented using a bit wise AND operation and the tree of adder, as shown in Fig 4.The final result of summation of tickets, T=r0t0+r1t1+r2t2+r3t3, defines the range for the random number to lie which was generated during procedure. The drawback of this dynamic lottery bus architecture is that the distribution of the resulting random number is not always same.[13,14].

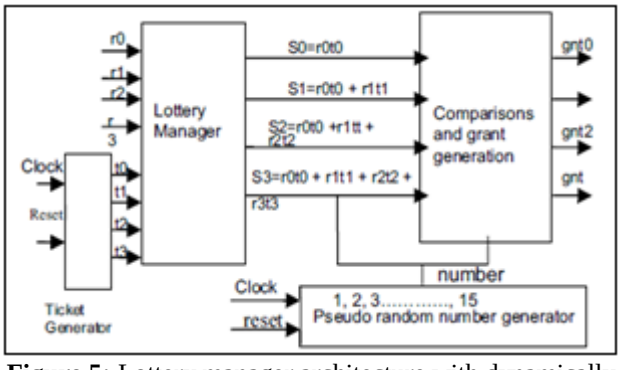

Figure 5: Lottery manager architecture with dynamically varying tickets

5. Results and Analysis

Following figure shows bus arbiter which is designed by dynamic lottery technique for four Users. Signal r0 r1 r2 r3 are the request signals from users and gnt0 gnt1 gnt2 gnt3 are the respective user grant signals. Signals t0 t1 t2 t3 represent the ticket value of the each respective user. Fig. 7,8,9 shows the simulation results for the Dynamic Lottery bus architecture.

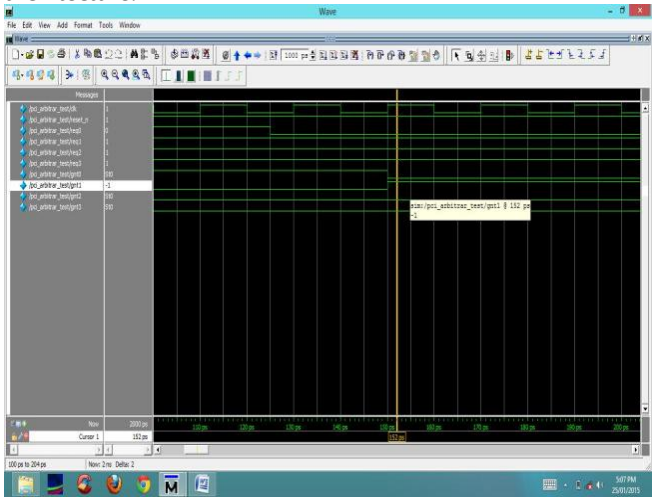

Figure 6: Simulation Result

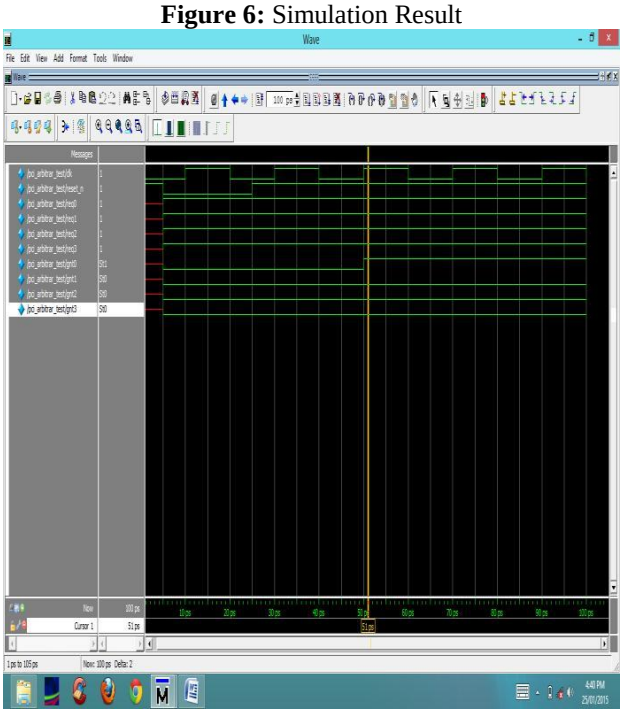

Figure 7: Simulation Result

Simulation results presented here in this paper are taken by Modelsim simulation software tool and found acceptable.

6. Conclusion

This paper presents Lottery bus arbitration technique for the interfacing with an arbiter and for the high performance SoC. Best arbitration technique is depends on the size and speed of the chip that is being built. Here In this paper, we propose the design of Lottery architecture by using efficient bus arbitration techniques like lottery bus communication architecture for high performance. The design of an arbiter managing handshake between clients and resources. Each resource is actively reporting for its availability and can be connected to any of the clients.

References

[1] G. Dent,D.J. "System –on-Chip research leads to hardware/software co-design degree", Frontiers in Educationa Conference,2000. FIE 2000.30th Annual Volume:2.988.
[2] K. Lahiri, A. Raghunathan, and S. Dey, "Performance Analysis of Systems with Multi-Channel Communication Architectures", *International Conference on VLSI design*. Jan 2000,pp,530-537.
[3] K. Lahiri, A. Raghunathan, and S. Dey, "System Level Performance analysis for designing on-chip communication architecture", *IEEE Trans. Computer-Aided* Des. Integr. Circuits Syst. Vol. 20. Jun 2001.
[4] K. Lahiri, A. Raghunathan, G, Lakshminaray, "LOTTERYBUS: A new high-performance communication architecture for system-on-chip Chang Hee Pyoun, et. all. "The Efficient Bus Arbitration Scheme In Soc Environment", IEEE International Workshop on System-on-chip, 2003.
[5] K. Lahiri, A. Raghunathan, ,"The LOTTERYBUS on-chip communication architecture", IEEE Trans. On VLSI system, June 2006.
[6] K.A Kettler, et.all, "Modeling Bus Scheduling Policies for Real Time Systems", 16th IEEE Real Time Systems Symposium, 1995.
[7] Vijay D'silva, S. Ramesh, Indian Institute of Technology Bombay, "Synchronous Protocol Automata: A Framework for Modelling and Verification of SoC Communication Architectures". Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'04) 1530-1591/04 $20.00 © 2004 IEEE.
[8] Hans-Joachim Solberg, "A Multi-Core System-on-Chip Architecture for Multimedia Signal Processing Applications"
Proceedings of the design, Automation and Test in Europe Conference and Exhibition (DATE'03) 1530-1591/03 © 2003 IEEE.
[9] Dinesh Padole, Deepsheekha , Dr Preeti Bajaj "Dynamic Lottery Bus Arbiter for Shared Bus System on Chip: A Design Approach with VHDL" First international conference on Emerging Trends in Engineering and Technology 2008 IEEE.
[10] Kanishka Lahiri and Anand Raghunathan, "Lotterybus: A new high-performance communication architecture for System-on-chip Designs", DAC 2001, June 18-22, 2001, ACM,USA.
[11] Tarun Kumar Gauttam, Rekha Agrawal, Sandhya Sharma," Arbiter Design Using Verilog for Switching to
[12] Communicate in Between Multiple Resources" International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-3, Issue-3, August 2013.
[13] Kanishka Lahiri and Anand Raghunathan,"Lotterybus: A new high-performance communication architecture for System-on-chip Designs", DAC 2001, June 18-22, 2001, ACM,USA.
[14] Dinesh Padole et.all, "Design and Performance analysis of efficient bus arbitration schemes for on-chip shared bus Multi-processor SoC", International Journal IJCSNS, Sept.08, Vol. 8 No. 9 pp. 250-255.