Implementation of AMBA Bus Arbiter using Multilevel AMBA AHB Round Robin and Fixed Priority Scheme

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

Abstract— On-chip bus architecture serves a very important role in on-chip design system. Now days there are many on-chip bus architecture design are provided by different company. One of the most popular is advance microcontroller bus architecture which is popularly known as AMBA 1.0. AMBA have a advantage that it is open specification i.e. AMBA serve as a framework for system on chip design. In On Chip System problem arise when number of master trying to sense a single data bus. Then resolutions of a required become an issue. The system performance depends upon a ability to resolve this resolution problem. The AMBA protocol use logical assignment of chance to different masters according to their priority to take over the bus for data transmission. AMBA architecture define three specification. They are advance high performance bus, advance peripheral bus and advance system bus. Among all three specification AHB have high bandwidth and this make AHB as first choice for system designer. While resolving the problem priority resolution, arbiter plays an important role. Arbiter is a digital circuit and working of it depends upon the arbitration algorithm. According to arbitration algorithm arbiter decide to give grant to master to access bus. In this project we design arbitration algorithm, according to it arbiter give grant to different masters. The device utilization of the proposed architecture is around 39% to 40%.xlinx simulator 14.5 vetex4 IC is use to implement and model sim simulator 10.1b is use for simulation.

Key words: 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.

1) Round Robin scheme
2) 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

Arbiter is a digital circuit. The main operation of arbiter is to grant access to master to shared resource. Arbiter block play an important role in SOC bus architecture. in SOC design many master are connected and they try to access bus. The problem arises when two or more master want to access bus at same time. Then arbiter decides which master grant to access bus and forces the other master to remain in ideal states. The process of choice or provide grant to master is according to arbiter algorithm .arbiter is important block in SOC design as it to efficiently handle the requests from all master and also responses from slave. Arbiter ensures that master requirement should be fulfilled. For example in some application master request real time or bandwidth requirement. then arbiter ensure that transaction accomplished with in fixed number of cycle for real time requirement and for fixed bandwidth requirement it ensure that master must occupy a fixed fraction of bandwidth of bus.

The arbiter algorithm could be implemented in two ways. There are centralized and distributed. In distributed arbitration algorithm slave side arbitration is absent where as in centralized arbitration algorithm it is present. To handle the configuration the arbitration algorithm must be optimized. In soc design for particular application power utilization by arbitration technique varies significantly

III. BLOCK DIAGRAM OF BUS ARBITER

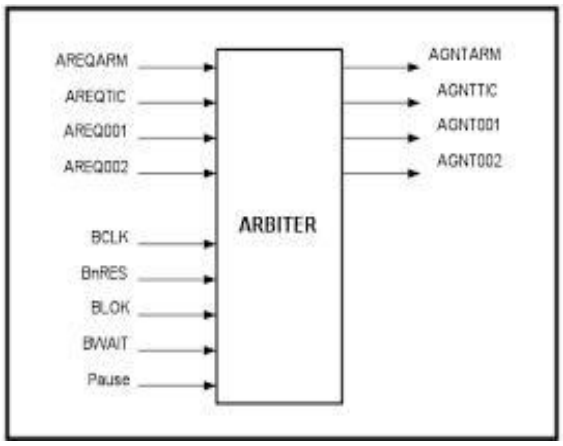

Fig. 1:

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

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 time is 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

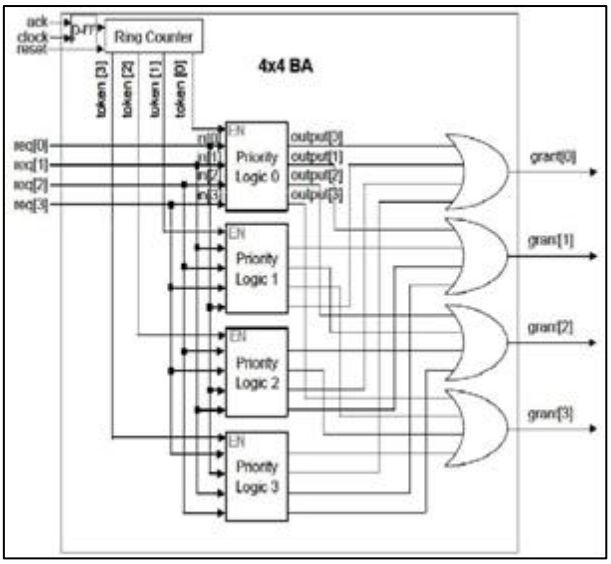

Fig. 2: Logic Diagram of 4x4 Bus Arbiter

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. PROPOSED ARBITER DESIGNS

Arbiter to choose master based on arbitration algorithm. Arbitration algorithm use round robin scheme to choose next bus master. Round robin arbitration is overridden when master has locked the bus and retain highest priority. For next bus master arbiter block monitors all bus request and choose high priority request if arbiter not find any request it give grant to default master to access the bus for next transaction. The controller which is mealy state machine keep track of all transaction of all different stations and it first state is start state and in next state it check the grant signal and if it is high then it make necessary signal high for further block interface.

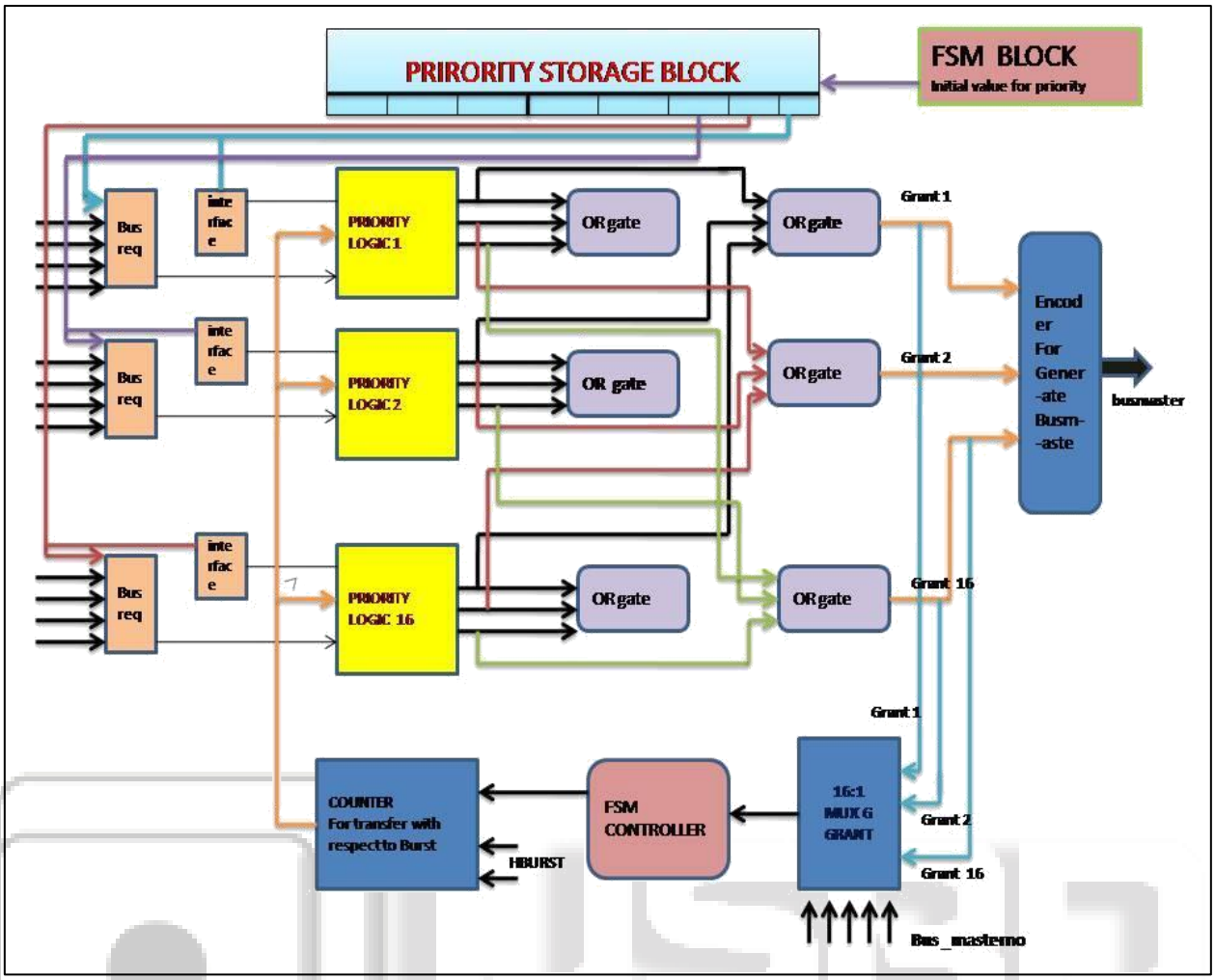

Fig. 3:

The bus-req blocks pass the entire master bus request through it to other logical block. The priority storage block enables the bus-req block. The priority logic is enable by interface block through enable signal also interface block is responsible for monitoring of data transaction, it monitor it by using data done signal and based result it assert and dessert the enable pin.

When priority logic block received bus request it further decided that which master have highest priority and give grant to it by generating grant signal. This grant signal interacts with master by passing through priority storage block, encoder block and out-put port, then master will send address, burst signal which define the type of transfer . After getting grant signal it response to it slave. Arbiter grant signal pass through multiplexer and bus master number will serve as select line for multiplexer which indicate master which accessing the bus. Then mux output is given to controller block which will generate necessary signal for counter block .The output grant signal from priority logic block is pass through OR gate and send to priority storage block .Depend upon the grant signal priority storage block pass the enable signal to next priority and depend upon the transaction mode ,whole operation repeated itself.

VI. 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).

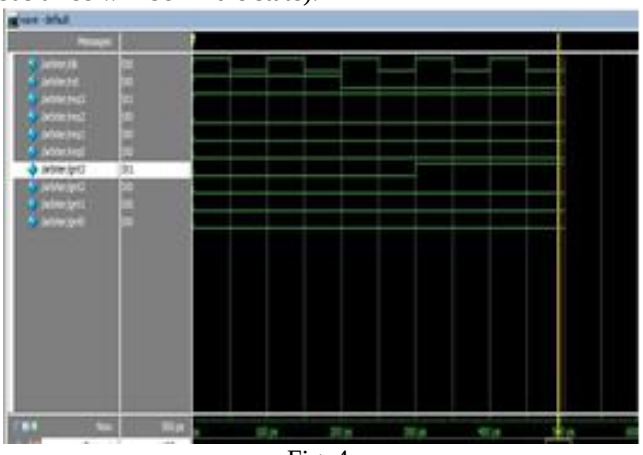

Fig. 4:

VII. Conclusion

AMBA AHB purposed arbiter is capable to perform arbitration process based on the combination of round robin and fixed priority algorithm. It is also capable to handle the request from sixteen masters

REFERENCES

[1] Multichannel AMBA ARB with Multiple Arbitration Technique by Shraddha Divekar, Archana Tiwari published in International Conference on Communication and Signal Processing, April 3-5, 2014 ieee
[2] Design Approach to Implementation of Arbitration Algorithm in Shared Bus Architectures (MPSoC) by D.Shanthi and Dr.R.Amutha published in Computer Engineering and Intelligent Systems Vol 2, No.4, 2011
[3] Implementation of Multilayer AHB busmatrix for ARM by E. Raja, K.V. Ramana published in International Journal of Soft Computing and Engineering, Volume-1, Issue-5, November 2011
[4] Implementation of Bus Arbiter Using Round Robin Scheme by Shashidhar R., Sujay S.N, Pavan G.S paper published in International Journal of Innovative Research in Science, Engineering and Technology Vol. 3, Issue 7, July 2014
[5] Design of an AMBA AHB Reconfigurable Arbiter for On-chip Bus Architecture by Pravin S. Shete, Dr.Shruti Oza published in International Journal of Application or Innovation in Engineering & Management Volume 3, Issue 5, May 2014
[6] Implementation Of An Adaptive-Dynamic Arbitration Scheme For The Multilayer Ahb Busmatrix by Dr. fazal noorbasha, b.srinivas, venkata aravind bezawada, v.sai Praveen paper published in International Journal of Engineering Research and Applications (IJERA) Vol. 2, Issue 4, July-August 2012, pp.825-831
[7] Slave-Side Arbitration and Implementation For Multilayer - AHB Bus Matrix by Mrs.R.Ramya, and Ms.Preethi Vadivel paper published in International Journal of Engineering Research and Reviews Vol. 2, Issue 3, pp: (56-60), Month: July - September 2014.

