ROUND ROBIN ARBITRATION DESIGN FOR DIFFERENT APPLICATIONS

PRIYA AMULRAJ, VENI N

Electronics and Communication Engg, SRM University, Kattankulathur Chennai, Tamil Nadu, India
a.darthipriyaa@gmail.com, veninkrishnan@rediffmail.com

ABSTRACT

This paper presents a Time efficient and an area efficient, two different ways of designing Round Robin Arbitration for different applications. An arbiter is a logical element which is helpful in selecting the order of access to a shared multi-master bus system. For each bus cycle, which multi-master bus system could be granted to control the bus is obtained using this logic. It uses priority logic and masking logic to decide which requester of many to be granted in a highly fair manner. This works in circular manner and thus starvation is avoided to maximum extent. The first method is suitable for applications where until the previously granted request has been de-asserted or reached its limit, it stays granted. Conversely, whenever the previously grant requests stays de-asserted, on priority basis requests are granted using masking concept. In real time it is used for applications like mobile phones where applications are chosen based on priority such that fairness is highly maintained. The second way of designing is suitable for applications where depending on the priority logic, one request by another is granted when multi-master bus system gives requests and only then moves on to deal with another input. Upon fetching another input it only grants request based on the previously masked output. In real time, it is used in computers where applications are chosen automatically. The Verilog code for both the ways are simulated using Quartus II software and are dumped in FPGA Cyclone IV for validation.

Index Terms— Area-efficient, Time efficient, Arbiters, Verilog, Quartus II

I. INTRODUCTION

In many days to day applications arbiters are necessary to overcome the complexity of selecting a request of many requests with strong fairness. Round Robin Arbiters are helpful in granting a request when 2 or more processors or applications requests access to a shared resource simultaneously. This arbiter grants the request depending upon the priority logic which is user defined. The arbitration works in a circular order. This arbitration logic is almost used in various fields of applications like communication, computers and so on. In this paper we are going to discuss about the design of two different ways in performing the arbitration using the masking and priority logic concept.

In method one, when multi-master bus system sends requests then only one request of many is being granted. It works with the logic that unless the previously granted request becomes passive i.e. gets de-asserted or reaches the user defined limit, it is signaled to be granted. Once it gets de-asserted in the next cycle then, based on the priority logic and masking concept the requests are granted. In implementing this way, it is to tell that the delay parameter is lesser when compared to other way of implementation.

In method two, when the multi-master bus system receives request it first grants the request one by one based on the priority logic in consecutive cycles and then it accepts the requests in next cycle. Meanwhile it updates itself with the masking concept which tells which requests are to be masked in the upcoming cycle and which all are allowed to be accepted. Here, the delay parameter is quite larger when compared to the previous method. But the area is being decreased comparatively.

When the system is busy in multitasking, this Round Robin Arbitration helps the system to avoid the collision and overcomes the bottleneck situation of granting a single request of many requests in a fair manner. Depending on the purpose of usage one of the two methods is used.

In this paper we are going to discuss about the design of the above mentioned two methods and the delay and area reports tabulation by using Quartus II soft-ware.

II. SYSTEM OVERVIEW

These arbiter circuits can be used in any system whenever the bizarre situation of selecting a request of many requests is to be solved. That is because the Verilog code is being written in generic manner so depending on the total available number of requesters the Verilog code would be made suitable accordingly.

The area is calculated based on the LUTs that is occupied. LUTs includes the flip flops and registers that is required to build the logic. The delay is based on the longest critical path that it occupies. The method one on implementation gives lesser delay with quite larger area occupation when compared to the method two and vice-versa.

The interfacing diagram for both the arbiter logics are shown in Figure 1. Method one and method two working flow is mentioned below in Figure 2 and Figure 3.

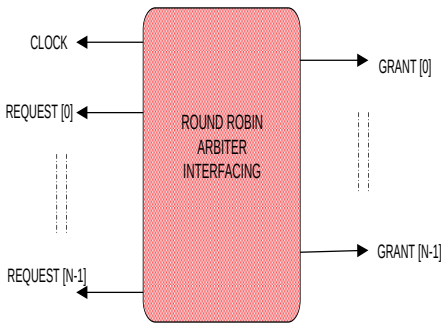

Fig 1- Interfacing diagram of RR Arbiter

A. Method One: This is based on the logic that when the request that is granted in previous cycle stays asserted in the upcoming cycles then, it is said to be granted until it gets de-asserted or had reached its limit. In my circuit I have considered *ack* signal as the

indicator signal to say if it had reached its limit or not. In which case previously grant request is granted till it becomes active low.

First, it checks if it has to grant the request based on the previously masked output logic or has to select the reset case. Then in every cycle the input requests are updated and it is being logically checked with the mask output using AND gate. This is then sent to priority logic unit which uses one hot encoding where only one request is made active high having set the remaining ones to be zero. Then it is sent to the masking unit where those requests that are served already are masked to zero and remaining are masked to one. If the currently served request is said to be *i* and there are in total *N* numbers of requests per cycle then *i*-1th to 0th requests are masked to zero and the remaining bits i.e. *i*th to *N*-1th requests are masked to one. The requests and its corresponding grant selection for *N*=4 is shown in Figure 2.

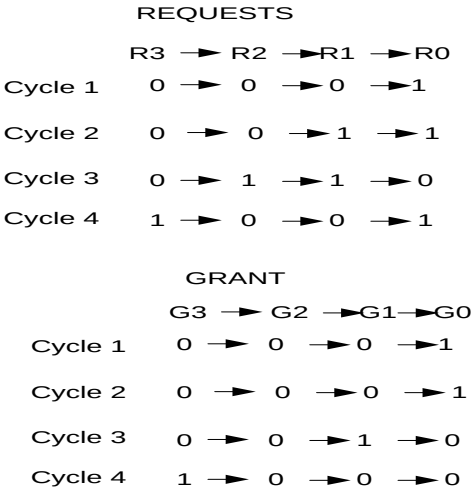

Fig 2- Method 1 request and grant selection

The working flow of the priority logic and masking concept as mentioned above is shown below in Table I and Table II

TABLE I

INPUT	OUTPUT
0000	0000
???1	0001
??10	0010
?100	0100
1000	1000

TABLE II

INPUT	OUTPUT
0000	1111
0001	1111
0010	1110
0100	1100
1000	1000

B. Method Two: This method is based on the logic that when multi-master bus system sends multiple requests then the requests are served one by one based on the priority logic and then moves onto masking concept and then based on this whenever it receives requests in next cycles it grants those requests based on the priority logic for those previously masked outputs. First, the request that is being granted in the previous cycle is sent to the masking unit from which we can get the masked output which uses an indicator to indicate whenever it is raised high, only then masking occurs else all the bits are raised high. Then this masked output is compared bit wise with the input requests using AND gate. Then this output is sent to the priority logic unit which grants an output request signal. The mask unit works as follows, if the currently served request is said to be *i* and there are in total *N* numbers of requests per cycle then *i*th to 0th requests are masked to zero and the remaining bits i.e. *i*+1th to *N*-1th requests are masked to one. The requests and its corresponding grant selection for *N*=4 is shown in Figure 3.

The working flow of the priority logic is the same as one briefed in Table I and masking concept as mentioned above is shown below in Table III

TABLE III

INPUT	OUTPUT
0000	1111
0001	1110
0010	1100
0100	1000
1000	1111

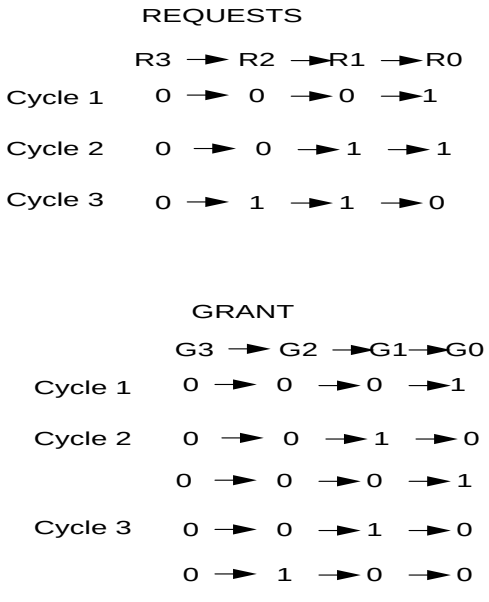

Fig 3- Method 2 request and grant selection

III DESIGN OF ROUND ROBIN ARBITER CIRCUIT

A. Method One: The design of the circuit based on the priority logic as well the masking concept is quite simpler but has little larger area occupation compared

to other method with lesser critical path delay. The block diagram and the circuit diagram is shown below in Figure 4(a) and Figure 4(b).

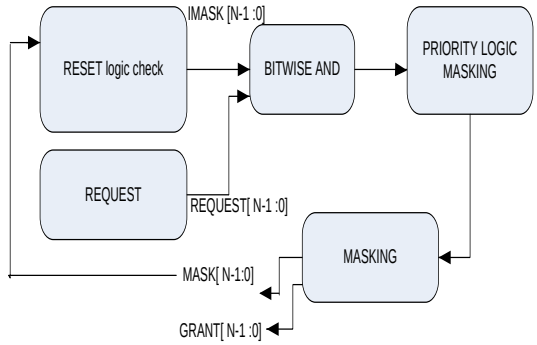

Fig 4(a) - Method 1 Block Diagram

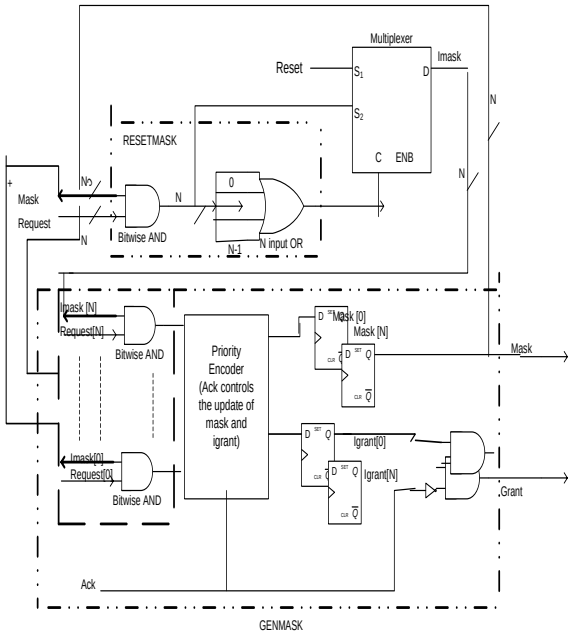

Fig 4(b): N- Bit Arbiter circuit for Method 1

The circuit diagram for the priority logic is shown below in Figure 5 whereas masking is done interms of mathematical expressions as given below in eqn (1).

mask [N-1:0] = (2 power N) - (2 power j) - (1)

where *N* is the total number of available requests and *j* is the location of the request that is granted in the previous cycle.

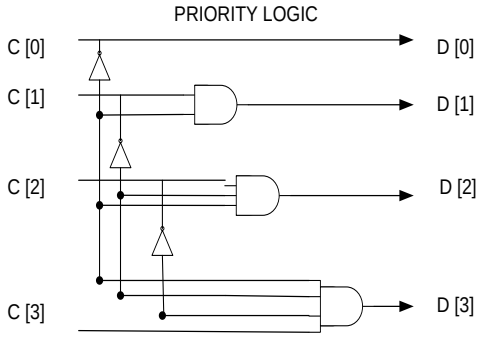

Fig 5: Priority Logic circuit

B. Method Two: The design of the circuit is shown below in Figure 6 and the masking circuit and priority logic circuit for the same design is shown in Figure 7 and Figure 8 respectively.

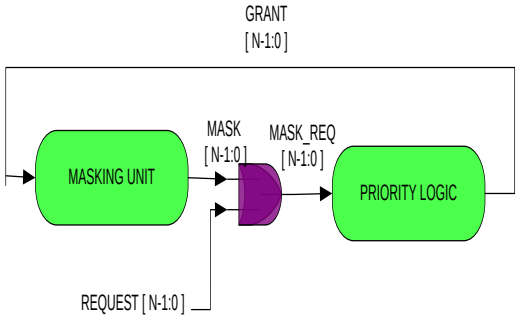

Fig 6 - Method 2 Block Diagram

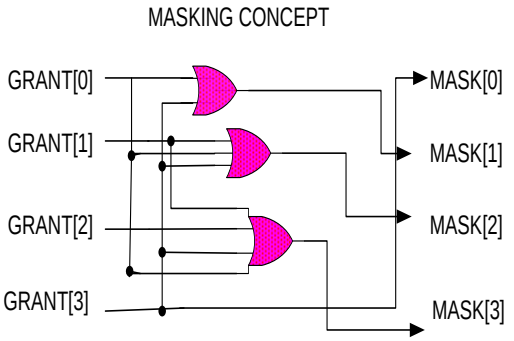

Fig 7 : Priority Logic

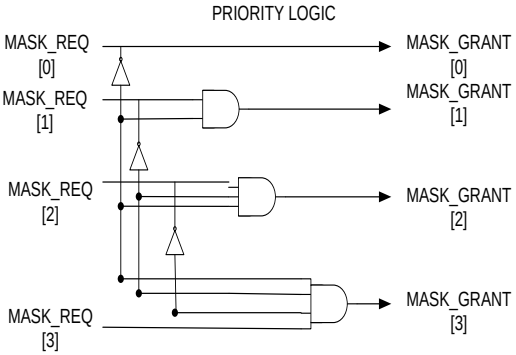

Fig 8 : Priority Logic

IV. EVALUATION OF ARBITER CIRCUIT

The evaluation parameters like Area and Delay for different numbers of requesters i.e. 4,8,16,32 for method 1 and method 2 are tabulated as follows in Table IV and Table V.

TABLE IV

NO OF REQUESTERS	AREA	DELAY
4	19	2.000
8	44	2.198
16	120	3.007
32	281	3.862

TABLE V

NO OF REQUESTERS	AREA	DELAY (ns)
4	10	2.000
8	22	2.525
16	52	3.320
32	107	4.350

As seen in the tabulation the area consumption in method 1 for having different *N* value is observed to be larger when compared to method 2. But it is the vice versa case for the delay calculation where the delay is observed to be lesser for method 1 compared to method 2 for different values of *N*.

So depending on the purpose and area of application we can use method 1 and method 2 as desired. The simulation results i.e. the waveforms after dumping in the FPGA (cyclone II) is shown in Figure 9 as follows.

The simulation results of both method 1 and method 2 are as shown in Figure 9 and Figure 10. The wave form of method 1 is obtained after dumping in FP -GA and similarly for method 2 it is done using Model Sim software.

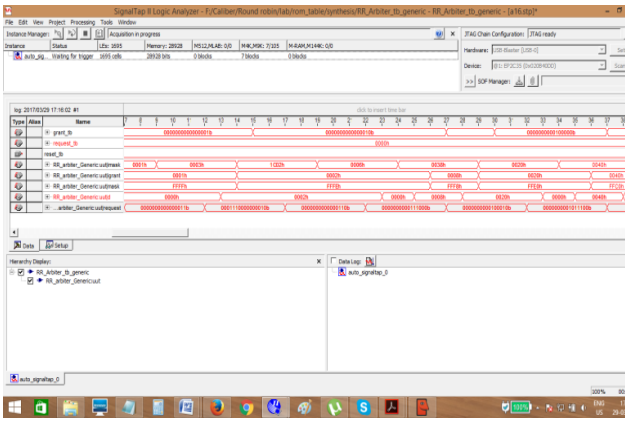

Fig 9: Simulated Results of method 1 after dumping in FPGA

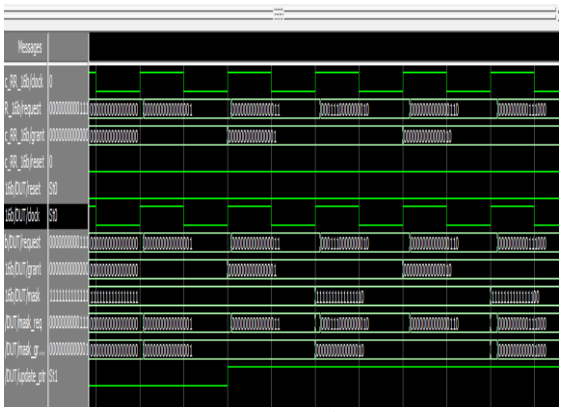

Fig 10: Simulated Results of method 2 using Model Sim

CONCLUSION

Implementing the Arbiter for any application and in bus systems based on the purpose in terms of delay and area. Masking and priority logics are used that are defined as per the requirement in circular manner. Round Robin Arbiter of Method 1 is that it grants a request of many requests that it receives based on the logic that whenever the previously grant requests stays assertive, it gets granted in upcoming signal. Similarly, whenever it gets de-asserted it checks for priority and uses masking concept in selecting the request to grant. Whenever the maximum limit is reached then it goes back to the reset case. Round Robin Arbiter of Method 2 is that it grants request one by one based on priority logic in a cycle and then corresponds to the next set of cycle by implementing masking concept. This also works by using one-hot masking and priority logic principles.

Advantage of method 1 is that the delay is minimum with quite larger area occupation whereas for method 2 the area consumption is minimum having quite larger. The results are verified using Quartus II software tool. This logic can be used for VLSI hardware implementation.

REFERENCES

[1] Pravin S. Shete, Shruti Oza, Design of an AMBA AHB Reconfigurable Arbiter for an On-chip Bus Architecture. IJAIEM 3(5): 245-252 (2014).

[2] Eung S. Shin, Vincent J. Mooney III and George F. Riley, Round-robin Arbiter Design and Generation, Georgia Institute of Technology, Atlanta, GA

[3] Asha S.N., Bindu A.U., Design and Implementation of Index Based Round Robin Arbiter for NOC Routers Using FPGA. IJIRSET 5(5): 7448-7453 (2016)

[4] http://www.rtlery.com/articles/how-design-round-robin-arbiter