Volume: 4 Issue: III Month of publication: March 2016
DOI:

www.ijraset.com
IC Value: 13.98

Volume 4 Issue III, March 2016
ISSN: 2321-9653

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Fair Round Robin: A Low Complexity Packet Scheduler with Proportional and Worst-Case Fairness

D.Amsa[1], K.K.Senthilkumar[2]

1 PG Scholar, ME (VLSI DESIGN), Mahendra Engineering College, Namakkal, Tamilnadu, India.
2 Assistant Professor, Department ECE, Mahendra Engineering College, Namakkal, Tamilnadu, India.

Abstract—Round robin arbiter (RRA) is a critical block in nowadays designs. It is widely found in System-on-chips and Network-on-chips. The need of an efficient RRA has increased extensively as it is a limiting performance block. In this paper,we deliver a comparative review between different RRA architectures found in literature. We also propose a novel efficient RRA architecture. The FPGA implementation results of the previous RRA architectures and our proposed one are given, that show the improvements of the proposed RRA.

Keywords: Packet scheduling, proportional fairness, worst case fairness, round robin scheduler

I. INTRODUCTION

The arbiter is a critical and widely used module in many SoC applications. When more than one agent, such as processing blocks, is sharing one resource, such as a channel, a buffer, or a switch port, an arbiter is used to assign this resource to one agent at a time. One of the most important features in an arbiter is the fairness, which represents how equally the resource is assigned to the different requesters. Therefore, the fairness is classified into three categories: _ Weak fairness: In which every request will eventually end up with a grant. Strong fairness: The average number of grants each requester will have is equal, when calculated over a sufficient number of arbitrations. _ FIFO fairness: The requesters get the grants in the order they request it.

Arbiters have many types that differ in fairness. The first type is the fixed priority arbiter in which the priority does not change over time; Instead, it is assigned linearly to the requester. It does not grantee any fairness; the highest priority requester can block other requesters. The second type is the variable priority arbiter. In this type, the priority changes from cycle to cycle to provide fairness. Variable priority arbiters are classified into oblivious arbiters and round-robin arbiters. In oblivious arbiters, the generated priority is independent of the last grant; instead, it depends on the last cycle priority. Hence, simple circuits, such as shifters, are used to generate the next priority, but oblivious arbiters give a weak fairness. Round- Robin Arbiters (RRAs) provide strong fairness by assigning the lowest priority to the last served requester. So the generated priority vector is a shifted version of the grant vector [1]. RRAs are broadly used in network routers. Network routers are responsible for redirecting packets from the sender node to the receiver node. They consist of crossbar switches, virtual channels and queues. The crossbar switch is responsible for connecting input ports with output ports. Many input ports may request to send packets to the same output port simultaneously.

So an arbitration circuit is needed by crossbar schedulers to decide which input port will be granted to pass to the output port. Crossbar schedulers are usually implemented using round-robin arbiter to avoid input ports starvation [2].In this paper, we study the performance of different architectures of RRAs in term of speed and area over FPGA, and then we propose a novel RRA architecture that provides significant improvements over previous architectures. The paper is organized as follows. Section II presents a literature review of different RRAs found in literature. The proposed arbiter is introduced in Section III, while the results are showed in Section IV. Finally, we conclude in Section V.

II. LITERATURE REVIEW

A. *Round-robin arbiter (RRA)*

In this section, we will review the different architectures of the round-robin arbiter (RRA) found in literature. Generally, the RRA consists of two main parts; the arbitration logic which generates the grant signals, and the priority update circuit which promotes the next input in line, after the last granted input, to the highest priority. A. Baseline arbiter The basic RRA consists of a group of cells, a cell for each request-grant pair, as shown in Fig. 1 [1]. Each cell keeps a priority flag to indicate that it has the highest priority.

Volume 4 Issue III, March 2016
ISSN: 2321-9653

Only one cell should be initialized with an active priority flag. The highest priority cell generates a grant signal corresponding to its request signal. If there is no request at this cell, the priority should be transferred to the next cell through the Carry-out signal (Cout). Carry-out signal of each cell is connected to the Carry-in (Cin) signal of the next cell, and the last cell's Cout is connected to the first cell's Cin so that the priority propagates from a cell to another. In case of no requests at all, the priority will propagate through all the cells back to the cell that has the old priority and the priority will not change.This wrap-around results in an undesirable combinational loop. Most static timing analyzers cannot properly analyze designs containing such loops. So synthesis tools are unable to optimize these designs.

B. *Timing speculative arbiter*

This arbiter limits the priority propagation to a fixed number of cells (s) [3]. If the priority needs to propagate longer than the (s) cells, the arbitration is pipelined. The propagation is limited by connecting each cell with the next (s) cell, i.e. a connection from the cell i to i+s. If there are no requests in the cells from i to i+s, the priority will be transferred to cell i+s directly in order to start the propagation from it in the next cycle. If no requests at all, the priority remains at the cell i. To distinguish between the pipeline case (requests beyond i+s) and the idle case (no requests), an extra signal (I) is added. (I) is the NOR of all input requests to detect an idle arbitration case. This architecture is faster than the baseline RRA but it still has a combinational loop.

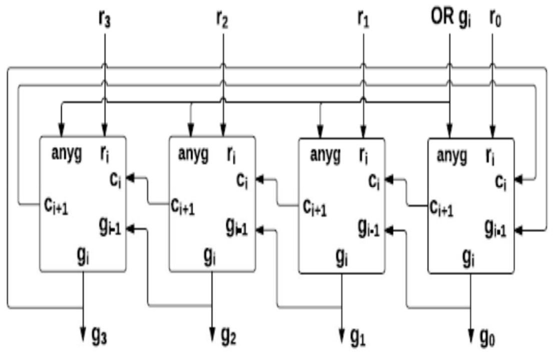

Figure 1: Baseline arbiter

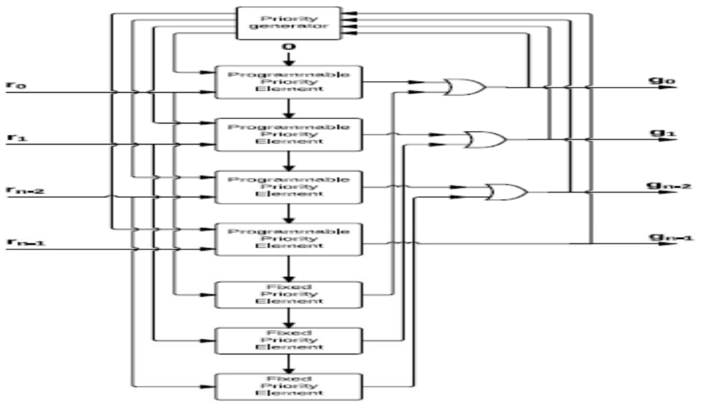

Figure 2: Acyclic Arbiter

C. *Acyclic arbiter*

To avoid the cyclic implementation, [4] proposed an acyclic RRA. In acyclic RRA, the wraparound between the last and the first cell is removed and it is replaced by a chain of fixed priority cells as shown in Fig. 2. The carry signals propagate the priority to the next cells if there is no requests at the higher priority cells. Instead of connecting the last carry signal with the first cell, the last cell propagates the carry to the fixed priority cells, in case it has the priority and has no request. This implementation avoids the combinational loop but it has a long critical path due to the carry propagation through both variable and fixed priority cells

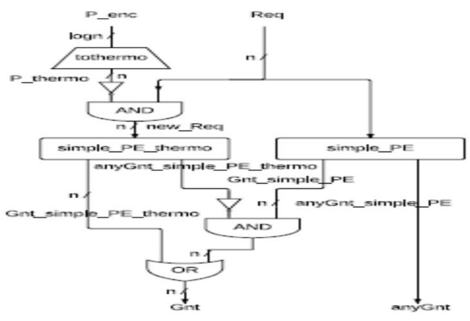

Figure 3: Dual-path PE Arbiter

.

www.ijraset.com
IC Value: 13.98
Volume 4 Issue III, March 2016
ISSN: 2321-9653

D. Priority-encoder based arbiter

The RRA can be built from priority encoders (PEs) [5]. A priority encoder is a fixed priority arbiter in which the first input has the highest priority. 1) Exhaustive PE arbiter: In this architecture, N PEs are used to make an N-bit RRA. The request vector is rotated and inputted to each PE, the rotation is done by wiring connections, so that each PE has a different request line on its first input, the highest priority position. Then a multiplexer selects one among these PEs according to the priority vector. This architecture requires large area of silicon especially for large values of N.

E. Dual-path PE arbiter

In [5], a dual-path PE architecture is introduced. It exploits two PEs in parallel to achieve roundrobin algorithm. The first PE searches for an active request starting from the highest priority bit to the last bit, it doesn't cycle back to the first bit. This is accomplished by disabling the requesters leading the highest priority request. This is done by coding the priority vector with a thermometer code and then masking it with the request vector. On the other side, the request vector is entered normally on the other PE without any masking. Both PEs work in parallel and then a multiplexer is used to choose between their outputs. The second PE is selected if there is no active requests after masking the request vector, so the multiplexer can be reduced as shown in Fig. 3. The priority update circuit is complex since the grant vector is in one-hot encoding and the priority is thermometer coded, so the grant vector needs to be thermometer encoded before updating the priority.

F. Parallel prefix arbiter

This architecture is proposed in [6]. Instead of using multiple fixed priority arbiters, this architecture generates each grant signal as a fully combinational function of all the bits of the request and the priority vectors. This architecture is fast but it
is not scalable for a higher number of requests.

III. PROPOSED ARBITER

Fixed priority arbiter has small area and high frequency over other arbiters, but it does not guarantee fairness between requests. On the other hand, round-robin arbiters consume more area and have a low operating frequency because of the priority propagation from location to another, but they do grantee fairness between requests.

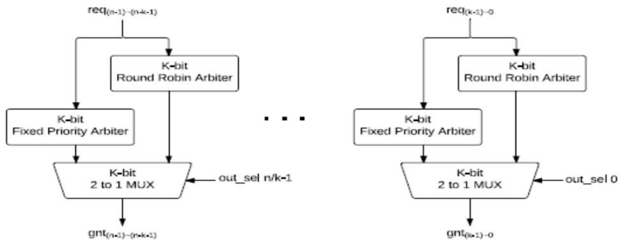

Figure 4: Priority-Select Arbiter

In our arbiter (priority-select arbiter), in order to elevate the operating frequency we divided the n-bit arbiter to n/k units, each unit consists of a k-bit round-robin arbiter and a k-bit fixed priority arbiter with a multiplexer to select between the two arbiters as shown in Fig. 4. The priority propagation direction is from the least bit to the most bit (from right to left) and back to the least bit again. When dividing n-bit requests to n/k groups, only one group (which contains the priority bit) needs to have a programmable priority, which we will refer to as the priority group. The remaining groups will have a fixed priority located at the least bit. Therefore, the multiplexer selects round-robin arbiters for the priority group while the remaining multiplexers select the fixed priority arbiters. Starting from the priority group, each group will block the following group's grants if it has at least one grant. For a k-bit round-robin arbiter, if the priority was at bit m, where 0_m_k-1, the bits i preceding m (0_i_m-1) should have a priority lower than all other group's bits. To do so, the priority of round-robin arbiter shouldn't propagate from the most bit to the least bit , which also enhances the frequency, hence the Round Robin arbiter will only respond to requests from the priority bit to the most bit. If there are any requests before the priority bit, and no requests in all other bits, all groups will generate no grants, including round-robin arbiter of priority group, in that case, the output of the fixed priority of the priority group will be selected.
For example, assume we have 24 bits arbiter (from bit 0 to 23) divided into 3 groups (from group 0 to 2), each of 8 bits as shown in Fig. 5, and the priority was at bit 12, and requests 9 to 11 were active. Group 1 will be the priority group and the multiplexer of

Volume 4 Issue III, March 2016
ISSN: 2321-9653

group 1 will select round-robin arbiter, while other groups multiplexer will select fixed arbiter. However, the round-robin arbiter of priority group will generate no grant, as the priority will not propagate from bit 15 to bit 8, and other groups also will not generate grants as they have no requests. In this case, the fixed arbiter of the priority group will be active and the grant bit 9 is set.

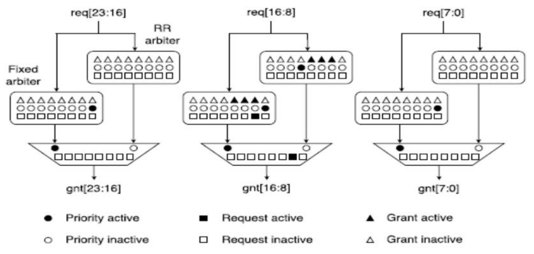

Figure 5: Priority-Select Arbiter Example

The k-bit round-robin arbiter can have any RRA architecture after simplifying it to remove the cyclic search, as we have mentioned that the carry propagation from the most bit to the least bit is not allowed. We choose the exhaustive PE architecture as the used k-bit RRA. The k-bit exhaustive PE RRA is fast but occupies a large area as it consists of k kbit PEs. But after removing the cyclic search, the k k-bit PEs are reduced to [k, k-1, k-2, ..., 2, 1]-bit PEs which implies significant decrease in the area used.

IV. RESULTS

In this section, the implementation results of the previous RRA architectures and our proposed architecture (priorityselect arbiter) are discussed. We have implemented all the different architectures stated in Section II, except the designs
which contain a combinational loop, among with the proposed one. The implementation is done on Virtex-5 FPGA (part xc5vlx50t-3ff665, speed grade -3). The implementation results are obtained by Xilinix ISE 14.6 tool. A summary of all implementation results is presented in Table I and Table II. Table I shows the area, in LUTs, and the maximum operating frequency, in MHz, of the previous architectures implemented on Virtex-5 for different number of requesters. The results assert that the parallel prefix and the exhaustive PE arbiters are not scalable, as their areas grow exponentially, but they have the highest operating frequency among the others. Table II shows the implementation results of the proposed arbiter for different number of requesters and different group size. From the table it is clear that the area increases gradually with the number of requesters, and provides the smallest area when compared with previous architectures. Moreover, the proposed arbiter preserves a high operating frequency that exceeds all previous architectures for a large number of requesters. The dominant delay in the proposed arbiter is the delay of the unit group, however, with increasing the number of arbiters, the critical delay moves to the control circuit which controls the selection of the unit group multiplexer, which leads us to an optimization problem that provides a degree of freedom to select the suitable group unit size for a given number of requesters.

	Area in LUTs					Frequency in MHz				
	n=4	n=8	n=16	n=32	n=64	n=4	n=8	n=16	n=32	n=64
Parallel Prefix	20	63	163	406	1031	505.8	449.8	316.1	292.1	242.8
Exhaustive PE	20	63	171	1453	3767	505.8	449.8	334.3	256.2	221.4
Acyclic	21	45	104	242	506	500.4	403.2	236.9	180.3	166.2
Dual-Path	7	49	115	262	500	740.1	256.9	184.3	143.0	137.3

Table I: Implementation Results of previous architectures for different no. of requesters (n)

	n=4	n=8		n=16			n=32				n=64				
	k=2	k=2	k=4	k=2	k=4	k=8	k=2	k=4	k=8	k=16	K=2	k=4	k=8	k=16	k=32
Area in LUTs	19	48	47	123	102	147	280	305	228	257	515	548	419	471	398
Frequency in MHz	493.4	378	329.9	268.1	244.3	286.2	209.8	223.9	284.843	248.7	193.5	200	251.4	215.1	216.6

Table II: Implementation Results of the proposed arbiter for different no. of requesters (n) and different bits per unit (k)

A comparison between our proposed arbiter, priority-select, and other arbiters is shown in Fig. 6 and Fig. 7. Fig. 6 shows the area of the implemented arbiters for different number of requesters. It shows that the proposed arbiter gives the lowest area among all the previous arbiters for large number of requesters. The configuration of the priority-select arbiter shown in the figure is as follows: k=2 for [4, 8]-bit arbiter and k=8 for [16, 32, 64]-bit arbiter (i.e. k is the group unit size). The maximum operating frequencies of the implemented arbiters are shown in Fig. 7. At large number of requesters, the proposed arbiter gives the highest maximum frequency. Both exhaustive PE and parallel prefix gives nearly the same performance but at the expense of area as shown in Fig. 6. The priority-select configuration is as follows: k=2 for [4, 8]- bit arbiter and k=8 for [16, 32, 64]-bit arbiter.

V. CONCLUSION

We deliver a comparative review between different round robin architectures found in literature. We also propose a novel efficient RRA which provides significant improvements over the previous RRA architectures. For a large number of requesters e.g. 64, the proposed arbiter gives the highest maximum frequency while occupying half the area of the nearest performance existing arbiter.

A. *Synthsis Report*

1. Device utilization summary:

Selected Device: 3s250etq144-4
Number of Slices: 84 out of 2448 3%
Number of Slice Flip Flops: 9 out of 4896 0%
Number of 4 input LUTs: 163 out of 4896 3%
Number of IOs: 20
Number of bonded IOBs: 20 out of 108 18%
Number of GCLKs: 1 out of 24 4%

B. *Timing Report*

Speed Grade: -4
Minimum period: 13.074ns (Maximum Frequency: 76.488MHz)
Minimum input arrival time before clock: 13.895ns
Maximum output required time after clock: 4.863ns
Maximum combinational path delay: No path found
Timing Detail:
All values displayed in nanoseconds (ns)
Total memory usage is 190712 kilobytes

VI. PERFORMANCE ANALYSIS

Three basic steps are involved in the design of an arbiter: modeling, logic verification, and synthesis. The design is first Modeled and may be represented in schematic graphs. The model is then described in Verilog HDL and the logic can then be verified by using simulation tool modelsim 6.3 version. Finally, the design can be mapped, to be synthesized and reduce delay and increase clock frequency. The synthesis tool used is XST synthesis, which optimizes a design constrained by timing. The same design can achieve a higher clock frequency. When comparing different design strategies of the same logic implementation, timing is used as the primary target in this project. As we see from comparison table the speed of the proposed work is being increased by 10.785 MHz in frequency and decrease in delay by 0.092ns

VII. CONCLUSION

The project on efficient speed implementation of round robin arbiter using verilog is been achieved. PPE arbiter design Using Islip algorithm which has much better performance than other practical algorithms under variant traffic models, without adding much complexity. Islip algorithm meets the criterion of a good scheduling algorithm good performance, fast and simple to implement. PPE arbiter design increases the system speed. It"s a much better approach than the static priority arbiter design. The project is been done using Xilinx ISE 9.1 with SPARTAN 3E family , XC3S500E device,FT256 package logically verified and then synthesized in XST synthesis tool and simulation in modelsim.6.3.using verilog HDL. The result obtained is increase in frequency by 10.78 MHz and delay is decrease by 0.092 ns. Thus the aim of the project is being achieved. The frequency of the Round Robin arbiter can further be increased if more better synthesis tool and simulation tool is being used such Synopsys.

Volume 4 Issue III, March 2016
ISSN: 2321-9653

REFERENCES

[1] W. Dally and B. Towles, Principles and Practices of Interconnection Networks. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.
[2] M. Abdelrasoul, M. Ragab, and V. Goulart, "Impact of round robin arbiters on router's performance for nocs on fpgas," in Circuits and Systems (ICCAS), 2013 IEEE International Conference on, Sept 2013, pp. 59–64.
[3] B. Zhao, Y. Zhang, and J. Yang, "A speculative arbiter design to enable high-frequency many-vc router in nocs," in Networks on Chip (NoCS), 2013 Seventh IEEE/ACM International Symposium on, April 2013, pp. 1–8.
[4] D. Becker, "Efficient microarchitecture for network-on-chip routers," Ph.D. dissertation, Stanford University, August 2012.
[5] P. Gupta and N. McKeown, "Designing and implementing a fast crossbar scheduler," IEEE Micro, vol. 19, no. 1, pp. 20–28, Jan. 1999.
[6] G. Dimitrakopoulos, N. Chrysos, and K. Galanopoulos, "Fast arbiters for on-chip network switches," in Computer Design, 2008. ICCD 2008. IEEE International Conference on, Oct 2008, pp. 664–670.
[7] N. Mckeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," IEEETransactions on Communications, 47: 1260-67, Aug. 1999.
[8] A. Mekkittikul, and N. Mckeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches," IEEE INFOCOM 98, San Francisco, April 1998.
[9] N. McKeown, "Scheduling Cells in an Input-Queued Switch," PhD thesis, University of California at Berkeley, May 1995.
[10] N. Mckeown, and T. E. Anderson, "A Quantitative Comparison of Iterative Scheduling Algorithms for Input-Queued Switches,"Computer Networks and ISDN systems, vol.30, pp. 2309-2326, 1997.
[11] H. J. Chao, and J.-S. Park, "Centralized Contention Resolution Schemes for A Large-Capacity Optical ATM Switch," Proc.IEEE ATM Workshop, Fairfax, VA, May 1998.
[12] D. N. Serpanos, and P. I. Antoniadis, "FIRM: A Class of Distributed Scheduling Algorithms for High-speed ATM Switches withMultiple Input Queues,"PROC. IEEE INFOCOM 2000, pp. 548-555. 2000.