ISSN ONLINE (2395-1680)
DOI: 10.21917/ijme.2015.0005

AREA AND DELAY MINIMIZED PROGRAMMABLE PREFIX ARBITERS FOR ON-CHIP COMMUNICATIONS

Viswanathan Nallasamy
Department of Electronics and Communication Engineering, Mahendra Engineering College, India
E-mail: vissivn@gmail.com

Abstract

Network-on-chip (NoC) is an effective on-chip communication technique; the core function of the crossbar schedulers used in the routers located into an NoC is arbitration which is required as and when a number of input ports of a router requests for a particular output port. The design of the arbiters is of paramount importance as the parameters like delay and area of the arbiters play a vital role in determining the performance of the NoC routers. In this paper, we present a circuit technique for the design of Programmable Prefix Arbiter (PPA) which is described in verilog and Modelsim simulator tool is used to validate the code. The study claims that average area (gate count) is reduced by 7%, propagation delay is decreased by 9% and operating frequency is increased upto 12% at the cost of 2% increase in the energy consumption in the design of PPA compared to that of the state of art Round-Robin Arbiter (RRA).

Keywords:

Network-on-chip, Router, Arbiters, Delay, Area

1. INTRODUCTION

Router is the backbone of an NoC interconnect architectures and its primary function is to forward each flit that arrives on one of its input ports to an appropriate output port [1]-[5]. A router is responsible for delivering packets effectively and reliably. It is a major component in an NoC that influences more on the performance and functionality of the NoC architectures. In the router design, input ports have First-In First-Out (FIFO) buffers in which incoming flits are stored temporarily until the flits are selected by the routers for further process [6]-[7]. The routers must be simple in design and fast in operation in order to meet the requirement of the on-chip communications [8].

Further, the routers must be implemented in an NoC by using an approach that minimizes chip area where NoC architecture is implemented, due to the fact that the area of a die per wafer of silicon is limited. The designers ought to target other parameters of latency and power constraints in addition to chip area while designing the routers required for a NoC. The router consists of four components which are input ports, crossbar switch, output ports and crossbar scheduler [9].

The crossbar scheduler is an essential component in a router which behaves like a brain of a router. The need for efficient implementation of simple crossbar schedulers has increased in the recent years due to the advent of on-chip interconnection networks that require low latency message delivery.

The primary operation performed by the crossbar scheduler is to mediate multiple input requests to access a shared resource and the act of coordinating the access is known as arbitration. A logic circuit implemented in a router to perform the function of arbitration is called as an arbiter.

An arbiter resolves the contention problem occurred during which more number of input ports requests for a particular output port. It grants one of multiple incoming requests to an output port based on the priority which depends on the crossbar scheduling algorithms. In an arbiter, grant signal is generated according to the input request signal and then the destination address is trapped. Finally, it generates select lines which are given to a crossbar switch to find out an output port.

The existing arbiter logic circuits are based on a symmetric implementation. The round-robin like algorithm requires one set of grant arbiters and another set of accept arbiters to perform the arbitration. The modified acyclic arbiter design removes one set of the arbiters and both the grant and accept arbitration are performed by using another set of arbiters in a time multiplexed fashion, thus the scheduler area can be minimized.

Fixed-priority arbiter is a basic arbiter circuit that requires pre-established priority order among the input requests [9], [10]. The priority of the arbiters has to be dynamic rather than fixed one so as to provide a fair allocation of resources and to achieve high performance router operation.

In recent years, the researchers contemplate more on designing the logic circuit for arbiters in which the priority has to be transferred from highest priority input port to next higher priority input port if the highest priority input port does not have a request for message transfer.

In this paper, arbiter logic circuit is designed, synthesized and implemented in Field Programmable Gate Array (FPGA); it is an acyclic arbiter in nature and has the capability to transfer priority signal if the highest priority input port does not have a request.

Remaining part of this paper is organized as follows: In section 2, design of the arbiter logic circuit is presented. Synthesis and implementation of the proposed logic circuit is explained in section 3. And the performance of the arbiter is analyzed in section 4. Finally, the conclusions are given in the last section.

2. DESIGN OF THE LOGIC CIRCUIT

In the proposed logic circuit design for an arbiter, the priority signal has to transfer from highest priority input port to higher priority input port. The priority transfer in an arbiter is realized using the concept of carry look ahead adder (CLA) [10], [11].

2.1 CARRY LOOK AHEAD ADDER

The carry bit c_i of a binary addition of two bits a_i and b_i at stage i is computed by using the equation,

$$c_i = g_i \vee (p_i \wedge c_{i-1})$$

where, generate bit $g_i = a_i \times b_i$ and propagate bit $p_i = a_i$ (xor) b_i and $c_0 = g_0$, $c_1 = g_1 + p_1 \times g_0$ and

$$c_2 = g_2 + p_2 \times c_1 = g_2 + p_2(g_1 + p_1 \times g_0)$$
$$= g_2 + p_2 \times g_1 + p_2 \times p_1 \times g_0, etc$$

CLA is implemented in three steps; the propagate and generate bits are computed in the first step; second step computes the carry generated for the stage i and sum of a binary addition is computed in the final step and the final sum is $s_i = p_i$ (xor) c_{i-1}.

Using the operator '∘', Boolean variables g_1, g_2 and p_1 and p_2 can be defined as follows:

$$(g_2, p_2) \circ (g_1, p_1) = (g_2 \vee (p_2 \wedge g_1), p_2 \wedge p_1)$$

The operator '∘' is an associative and hence CLA can be computed in parallel.

2.2 THREE BITS PROGRAMMABLE PREFIX ARBITERS

In the proposed logic circuit design, the carry generate bit g_i is replaced by the priority signal p_i which is also called as priority generate bit and instead of the carry propagate bit p_i, the inverted input request signal R_{i-1} is used. The priority transfer signal X_i is computed by using the signals p_i and $\overline{R_{i-1}}$ as follows. The signal X_i is used to transfer the priority signal from highest priority input to next higher priority input if the highest priority input does not have a request [9], [12], [13].

The grant signal G_{ri} is generated by multiplying the input request signal R_i with the priority transfer signal X_i.

The required three priority transfer signals X_0, X_1 and X_2 are computed as follows:

$$X_0 \leftrightarrow (P_0, \overline{R_2}) \circ (P_2, \overline{R_1}) \circ (P_1, \overline{R_0})$$
$$X_0 = P_0 \vee (\overline{R_2} \wedge (P_2 \vee (\overline{R_1} \wedge P_1), \overline{R_2} \wedge \overline{R_1} \wedge \overline{R_0})$$
$$X_0 = P_0 + \overline{R_2} \times P_2 + \overline{R_2} \times \overline{R_1} \times P_1$$
$$X_1 \leftrightarrow (P_1, \overline{R_0}) \circ (P_0, \overline{R_2}) \circ (P_2, \overline{R_1})$$
$$X_1 = P_1 + \overline{R_0} \times P_0 + \overline{R_2} \times \overline{R_0} \times P_2$$
$$X_2 \leftrightarrow (P_2, \overline{R_1}) \circ (P_1, \overline{R_0}) \circ (P_0, \overline{R_2})$$
$$X_2 = P_2 + \overline{R_1} \times P_1 + \overline{R_0} \times \overline{R_1} \times P_0$$

The logic circuit implementation for the three bit arbiter is shown in Fig.1 in which the pairs $(P_2, \overline{R_1}), (P_1, \overline{R_0})$ and $(P_0, \overline{R_2})$ are selected in order to ensure the cyclic nature of the priority transfer signal. The term 'prefix' can be defined as the outcome of an operation depending on the initial inputs. Any arbitrary primitive operator used in an operation is associative then the operation can be executed in parallel [14], [15].

The parallel operation is fast because the processing is carried out in a parallel fashion. In the logic circuit shown in Fig.1, the Boolean operations are performed in parallel and the transfer of the priority signal is dynamic and hence the arbiter is called as three bit Programmable Prefix Arbiter (PPA). Table.1 shows the partial truth table of the three bit PPA. It is noted from Table.1 that the priority is transferred from P_1 to P_0 since the input requests R_1 and R_2 are inactive and R_0 has an active request.

The priority is transferred from P_2 to P_0 while the input requests R_0 and R_1 are active and R_2 is inactive; Further the priority is transferred from P_2 to P_1 as the input request R_1 is active and R_0 and R_2 are inactive.

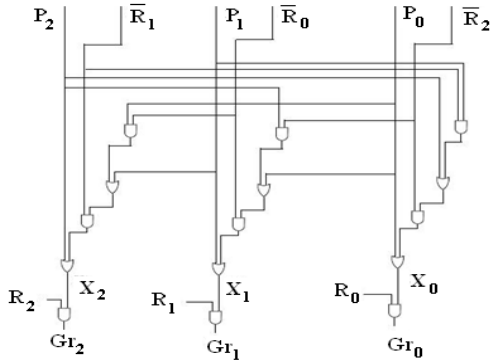

Fig.1. Circuit implementation for three bit PPA

Similarly, the priority transfer signals for four, eight and sixteen input/output ports arbiters are computed and required logic circuits are realized.

Table.1. Partial truth table for three bit PPA

Priority Signal			Request Signal			Grant Signal		
P_0	P_1	P_2	R_0	R_1	R_2	G_{r0}	G_{r1}	G_{r2}
1	0	0	1	0	0	1	0	0
0	1	0	1	0	0	1	0	0
0	0	1	1	1	0	1	0	0
0	1	0	1	0	1	0	0	1
1	0	0	0	1	1	0	1	0

The Fig.2 shows the block diagram for the design of sixteen port arbiter; three input signals R, $\overline{R}$ and P are fed into the arbiter and the priority transfer signal X is computed to generate a grant signal G_r.

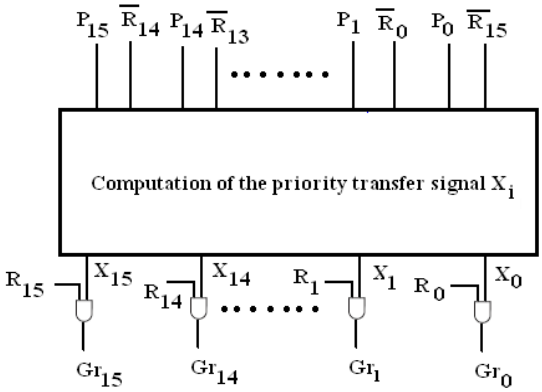

Fig.2. Block diagram for the design of sixteen port PPA

The arbiter with two, four, eight and sixteen ports is synthesized using a synthesis tool in which 3E-XC3S500E-5FT256C is the target technology used [15].

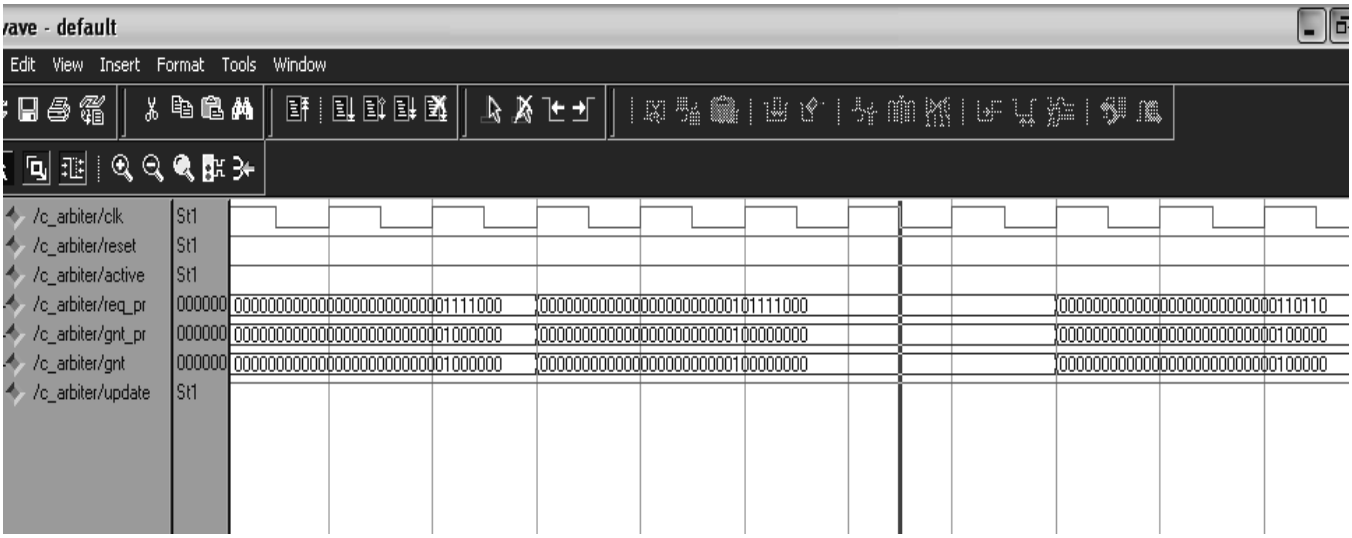

Fig.3. Screen capture of the waveform for sixteen ports PPA

3. SYNTHESIS AND IMPLEMENTATION OF PPA

The circuits designed in the previous section for realizing the PPAs are synthesized in Xilinx ISE 9.2i. Source code for realizing the circuits are written in Verilog HDL.

Modelsim simulator is used to simulate the arbiter and the screen capture of the simulation output is shown in Fig.3. In the Fig.3, arbiter/req-pr and arbiter/grt-pr are the sixteen ports PPA input and output wave forms respectively. Further, more than one request is applied to the arbiter through sixteen input ports 'arbiter/req-pr' at three different instances; the grant signals 'arbiter/grt-pr' generated from the arbiter at the three different instances are shown in Fig.3.

It is observed from the wave forms that only highest priority input request is granted at every instance. The priority is transferred from highest to next higher priority port if the highest priority port request is inactive. The PPA circuit is implemented in Xilinx FPGA Spartan 3E-XC3S500E-5FT256C kit to verify the operation of the PPA using both simulation and hardware implementation. The output port selection for various requests of input ports of a sixteen ports arbiter is observed in the kit and verified the results with the results of the Modelsim simulator output as shown in Table.2.

Table.2. Output port selected at various instances in a sixteen port arbiter

Number of Instances	Requests for input ports	Output port selected by a sixteen port arbiter
1	0000000001111000	0000000001000000
2	0000000101111000	0000000100000000
3	0000000000110110	0000000000100000
4	0000000010101010	0000000010000000
5	0000000011111101	0000000010000000

The scaling behavior of the arbiter is evaluated by selecting the arbiter with two, four, eight and sixteen ports. The number of input / output ports of the arbiter is selected based on 2^n where n = 1, 2, 3 and 4. Round Robin Arbiter with prefix network (RRA) used in [16] is synthesized by using the same synthesis tool and device used for implementing the PPAs.

4. PERFORMANCE ANALYSIS OF PPA

In this section, the performance of the arbiter with two, four, eight and sixteen ports is analyzed in respect of area (gate count), delay and energy consumption and compared with RRA.

The experimental results of PPAs and RRAs are shown in Table.3 and it is observed from Table.3 that the average area (gate count) occupied by the arbiters is reduced by 7% and propagation delay is reduced by 9% in PPA compared to that of RRA. The reduction of area (gate count) and delay in PPA is observed due to 7% lesser number of gates used in the circuit design and 12% increase in operating frequency.

Further, an average 2% increase in energy consumption is observed in PPA than that of RRA due to the increase in the operating speed of the process. Hence it is concluded from the analysis that PPA outperforms RRA in respect of area (gate count), delay and operating frequency at the cost of 2% increase in energy consumption.

PPA and RRA with sixteen input / output ports are implemented in the routers / switches of the two topologies used in [7]. The performance and cost metrics of the two topologies in respect of chip area, delay and energy consumption are analyzed under the two traffic patterns hot spot and nearest neighbor.

It is observed from the analysis that an average of 5% area (gate count) required for implementing the arbiter and an average of 8% propagation delay are reduced while PPA is implemented in the routers / switches of the topologies in place of RRA at the cost of 1% increase in energy consumption. The increase in the energy consumption is due to 7% increase in the operating frequency of PPA than that of RRA.

Table.3. Performance comparison of PPA with RRA

Parameters	RRA Port -2	PPA Port -2	RRA Port - 4	PPA Port - 4	RRA Port - 8	PPA Port -8	RRA Port -16	PPA Port -16
Number of Slice Flip Flops	1	1	2	2	3	3	7	4
Number of 4 input LUTs	3	3	21	19	66	58	144	143
Number of occupied Slices	2	2	11	10	35	30	77	75
Gate count for design	29	29	154	139	432	381	944	899
Max. Frequency in Mhz	467	467	163	241	90	117	75	80
Max. Delay in ns	6.34	6.34	8.68	7.85	11.49	9.83	12.31	11.37
Power Consumption in mW	104	104	93	95	92	95	96	97

5. CONCLUSION

A Programmable Prefix Arbiter (PPA) is designed, synthesized in Xilinx ISE 9.2i and implemented in Xilinx FPGA Spartan 3E-XC3S500E-5FT256C kit. The performance of the arbiter with two, four, eight and sixteen ports is studied and the performance of PPA is compared with the state-of-art Round Robin Arbiter with prefix network (RRA). In the arbiter design, the experimental results show that the average area (gate count) occupied by the arbiters is reduced by 7% and the propagation delay is reduced by 9% in PPA compared to that of RRA. The reduction of the area and delay in PPA is observed due to 7% lesser number of gates used in the circuit design and 12% increase in operating frequency. Further, 2% increase in energy consumption is observed in PPA than that of RRA due to the increase in the operating speed of the process. It is observed from the analysis that PPA outperforms RRA in respect of area (gate count), delay and operating frequency at the cost of 2% increase in energy consumption. The results and analysis of the present work is very much useful for theoretical study and practical implementation of routers in a NoC interconnect architecture.

REFERENCES

[1] L. Benini and G. De Micheli, "Networks on chip: A new SoC paradigm", *IEEE Computers*, Vol. 31, No. 1, pp. 70-78, 2002.

[2] W. J. Dally and B. Towles, "Route packets, not wires: On chip interconnection networks", *Proceedings of Design Automation Conference*, pp. 683- 689, 2001.

[3] A. M. Rahmani, K. Latif, P. Liljeberg, J. Plosila and H. Tenhunen, "Research and Practices on 3D Networks-on-Chip Architectures", *Proceedings of NORCHIP Conference*, pp 1-6, 2010.

[4] A. Agarwal, C. Iskander and Ravi Shankar, "Survey of Network on Chip (NoC) Architectures & Contributions", *Scientific Journals International*, Vol. 3, No. 1, pp. 1-15, 2009.

[5] Sheng Ma, Natalie Enright Jerger and Zhiying Wang, "Whole Packet Forwarding: Efficient Design of Fully Adaptive Routing Algorithms for Networks-on-Chip", *Proceedings of the 18th International Symposium on High-Performance Computer Architecture*, 2012.

[6] F. Dubois, A. Sheibanyrad, F. Petrot and M. Bahmani, "Elevator-First: a Deadlock-Free Distributed Routing Algorithm for Vertically Partially Connected 3D- NoCs", *IEEE Transactions on Computers*, Vol. 62, No. 3, pp. 609-615, 2013.

[7] N. Viswanathan, K. Paramasivam and K. Somasundaram, "An Optimized 3D Topology for On-Chip Communications", *International Journal of Parallel, Emergent and Distributed Systems*, Vol. 29, No. 4, pp. 346-362, 2014.

[8] P. Gupta and N. McKeown, "Design and implementation of a fast crossbar scheduler", *IEEE Micro Magazine*, Vol. 19, No.1, pp. 20-28, 1998.

[9] E. S. Shin, V. J. M. III and G. F. Riley, "Round-robin Arbiter Design and Generation", *Proceedings of the International Symposium on System Synthesis*, 2002.

[10] G. Dimitrakopoulos, H. T. Vergos, D. Nikolos and C. Efstathiou, "A family of parallel prefix modulo 2n-1 adders", *Proceedings IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pp. 326-336, 2003.

[11] Daniel U Becker and William J. Dally, "Allocator Implementations for Network on-Chip Routers", *Proceedings of ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1-12, 2009.

[12] Wladek Olesinski, Hans Eberle and Nils Gura, "PWWFA: The Parallel Wrapped Wave Front Arbiter for Large Switches", *Proceedings of IEEE Work shop on High Performance Switching and Routing*, pp. 1-6, 2007.

[13] Gupta Pankaj and Nick Mckeown, "Designing and Implementing a Fast Crossbar Scheduler", *IEEE Micro*, Vol. 19, No. 1, pp. 20-28, 1999.

[14] Suyog K Dahule and M. A. Gaikwad, "Design & Analysis of Matrix Arbiter for NoC Architecture", *International Journal of Advanced Research in Computer Science and Electronics Engineering*, Vol. 1, No. 5, pp. 100-103, 2012.

[15] Giorgios Dimitrakopoulos, Nikos Chrysos and Kostas Galanopoulos, "Fast Arbiters for On-Chip Network Switches", *Proceedings of IEEE International Conference on Computer Design*, pp. 664-670, 2008.

[16] Daniel U Becker, "Efficient Micro-architecture for Network-on-Chip Routers", Ph.D Thesis Dissertation, Stanford University, 2012.