Synthesis & Evaluation of Different Allocation Algorithms on FPGA

Asha S N

Assistant professor
Electronics and Communication Engineering,
BGS College of Engineering and Technology, Mahalakshmipuram, West of Chord Road, Bengaluru -560 086, Karnataka.

***Abstract*—Network on Chip (NoC) is an efficient solution to handle distinctive challenges by providing scalable communication infrastructure among on-chip resources. Arbiter is used in Network on Chip (NoC) when number of inputs are requested for same output port, the arbiter needs to generate the grant signal on the basis of priority assigned to the input port.Arbiter generates this grant signal depending on allocation algorithm. The arbitration algorithm plays an important role in determining the perforemce of the whole NoC based mesh.The proposed project revaluates some of the allocation algorithms like iSLIP Allocator, Wavefront Allocator, Lonely Output Allocator and Wormhole Allocator and presents the synthesis and implementation on FPGA platforms. The work will help NoC designers to choose allocation algorithms for their FPGA design. The parameters considered are resource utilization, timing and average power dissipation. Resource utilization is considered in terms of on-chip FPGA components used. A Fair comparison of the result is obtained using test bench with same clock period and same input statistics in all topologies implemented. The implementation and design synthesis is carried out in Xilinx ISE 14.2. The operating frequency and clock period of the implemented design is observed in simulator data base. Power metrics are measured using Xpower analyzer.**

I. INTRODUCTION:

A variety of Interconnection schemes are currently in use, which includes buses, crossbars and NoC's. In research community buses and NoC's are dominant. However buses are not scalable because their performance decreases as number of processing elements increases. So the buses are not used where processing elements are large in number. To overcome this poor scalability limitation, packet based on-chip communication network known as NoC (Network on chip) are used.

In past few years, NoC has got lot of attention by providing higher bandwidth and higher performance architectures for communication on chip. NoC's (Network on chip)are communication subsystem on an Integrated Circuits. If NoC's are implemented on reconfigurable platforms they provide simple and scalable architectures. SOC (System on chip) which contains many processing elements are connected through NoC router. These Network on chip routers are arranged in regular fashion such as Torrous. Mesh, linear, 2D, 3D type of topologies. Figure 1 illustrates some of these topologies.

A router should provide high bandwidth and low latency to achieve high performance. The throughput of Network on chip depends on network topology, router throughput and the traffic load at the network. The performance of Network on chip depends on these characters. Throughput of router is determined by critical path of data path units in the router and the efficiency of control path units. Router plays an important role in on-chip network which under takes crucial task of coordinating the data flow.

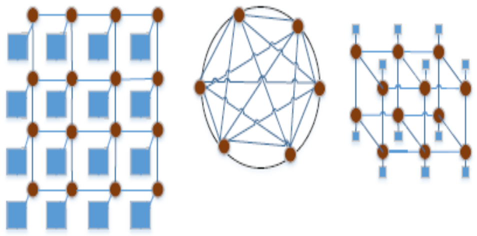


Figure 1.1: Block diagram of various NoC topologies.

The on-chip router consists of buffers, VC (Virtual channel) and switching fabric. The control path of on-chip communication router consists of arbiters and allocators. To allocate virtual channels and to perform matching between resources allocators are used, while arbiter is used to allocate access to shared resources.

A switch allocator must match the input units and the output ports of router so that at least one flit/cell from each input port is selected and at least one flit/cell destined to each output port is selected. The project focuses on implementation of various algorithms for allocator. The port density selected is 12bit request vector. The project introduces implementation and synthesis of various allocation algorithm used in allocator.

II. RELATED WORK:

The Report presented by the author introduces a wormhole switch micro architecture. The flit admission and flit ejection are problems that concern about how flits of packets are admitted into and ejected from the network. A novel coupling scheme used for flit admission which binds a flit admission queue with an output physical channel. The paper shows that the above scheme achieves reduction of up to 8% in switch area and up to 35% in switch power over the comparable solution. Another scheme referred to p-sink model is ejection. [1]

In NoC each router has responsibility of resource allocation. Distributed nature of NoC gives advantage of scalable performance of NoC but complicates the problem of providing Quality of service (QoS) support to individual flows. Problems such as low network utilization and weak Quality of service (QoS) occur in existing approaches. [2]

The paper presents a router known as TS router from the demonstration that the matching decisions made by a router along time forms a time series and Quality of allocation (QoA) is maximized if matching decisions is made across the time series, to refuse requests from past history. TS router predicts future requests that can arrive at a router and it tries to maximize the match across cycles. [3]

The paper proposes a dynamic batch co-scheduling (DBCS) scheme to schedule packets in a heterogeneous network processor by assuming that the work load is divisible perfectly. The author analyzed derive expression and throughput for scheduling time, batch size and maximum number of schedulable processor. To schedule variable length packets effectively in network processors (NPs), a packetized dynamic batch co-scheduling (P-DBCS) scheme with a combination of deficit round robin (DRR) and surplus round robin (SRR) schemes are proposed. The algorithm is extended to handle multiple flows on fair scheduling of flows depending on reservations.[4]

III. SYSTEM DEVELOPMENT:

3.1 Wormhole Allocator:

In general Wormhole is a hypothetical topological feature of space time which is 'short cut' through space and time. Travelling through Wormhole takes less time than travelling between the same distances in normal space.

Wormhole switching is one of the widely used switching mechanisms for on-chip network. In Wormhole networks, Flit is the smallest flow control unit. Large data packets are encapsulated into smaller Flits. These Flits are transmitted in a pipeline fashion. Using Flits small network delivery time is achieved and is advantageous over store-and- forwarding switching. It uses smaller buffers because it stores Flits instead of packets. These Flits are then admitted into and delivered in the network. Figure 3.1 shows how flits are stored and transmitted to the destination.

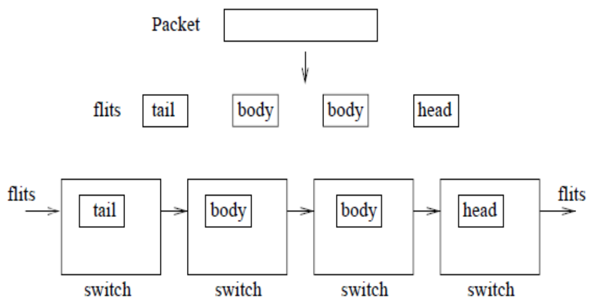


Figure 3.1 Flits delivered in a pipeline

Wormhole switching is sometimes called cut-through-switching because it cuts large sized packets into smaller units called Flits while virtual cut-through switching requires storing packets. In wormhole switching large network packets are broken into smaller units called FLITS. A Flit contains header flit which holds the information about its destination address.

The head Flit is followed by body Flit which contains the data to be transmitted to the destination. The last flit is called as Tail Flit which indicates the close of connection between the two nodes. Wormhole implements virtual channels to control the flow of Flits. Figure 3.2 shows data encapsulated in packet and Flit.

Virtual channels define the state required to handle the Flits of a packets over a channel. Wormhole routing is advantageous as it provides lower latency compared to other routing schemes. It uses smaller Flit buffers at nodes which reduces cost and area. Channels can be shared by many packets by virtual channels (VC). Virtual Channels in Wormhole routing refers to a pair of Flit buffers in nodes connected by shared physical channel. This physical channel is time shared by all virtual channels (VC).

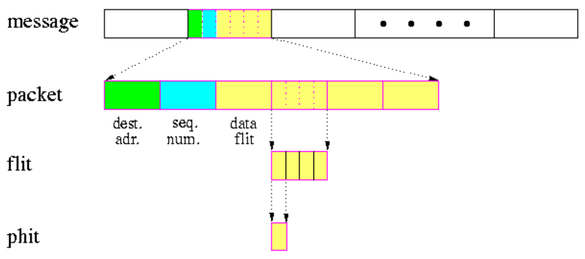


Figure 3.2: Data Encapsulation

But there is disadvantage of causing deadlock i.e., a situation where two or more competing actions wait for each other to compete, and thus the process of wait never ends. To avoid deadlock situation preemptions of packets should be allowed and the preempted packets must be discarded.

Instead of packets, Flits are allocated with buffers and physical Channels (PC) in Wormhole Switching. The process of dividing packets into Flits is referred as Flitization. Sometimes packets get blocked leading to blocking of all physical channels (PC) held by these packets. To avoid this problem and to make efficient use of physical channels (PC), Virtual channels are used. Several parallel lanes, each of which is a flit buffer queue, share a Physical Channel (PC). This causes efficient utilization of Physical channel (PC) giving high throughput because whenever a packet gets blocked other packets can still transverse the Physical channel (PC) through other lanes.

Wormhole allocation uses handshaking process where adjacent nodes communicate each other before transmitting and receiving flits. They communicate using a one bit ready/request(R/A) line for request and acknowledgments. When senders want to transmit the data and are ready. Then (R/A) line is high and Flit is transmitted. When a receiver is ready, the (R/A) line is low. After the receiver receives the Flit (R/A) is lower which indicates that it is ready to accept another Flit. This process is called handshaking and the cycle repeats for transmission of other Flits.

Latency of storing and forwarding the data packets is L/W*(D+1) while Wormhole latency is L/W+F/W*D
Where, D – number of intermediate nodes between source and destination
L – Packet length (in bits)
F – Flit length (in bits)
W – Channel bandwidth (in bits/sec)
F<<D, hence intermediate nodes have no significant effect on latency in wormhole systems.
Wormhole referred as Flit-buffer flow control with Buffers allocated per Flit. A head Flit in Wormhole flow control must acquire three resources at the next switch before its being forwarded:

- Channel control state i.e., virtual channel, one per input port
- One Flit buffer
- One Flit of channel Bandwidth

The other Flits adopt same virtual channel has the head and only compete for the buffer and the physical channel.

3.2 iSLIP Allocator:
iSLIP allocation algorithm is an iterative round robin algorithm. To select a cross bar configuration, multiple iterations are performed at every time slot i.e., to match inputs to outputs. iSLIP algorithm works in round robin fashion which refers to rotating priority. Each active input is scheduled based upon its round robin priority by the arbiter.

iSLIP scheduling algorithm is simple, can operate at high speed and can be easily implemented in hardware. iSLIP achieves 100% throughput by making conflict free matching in multiple iterations. Figure 3.4 shows Implementation of Arbiter in iSLIP Scheduling algorithm.

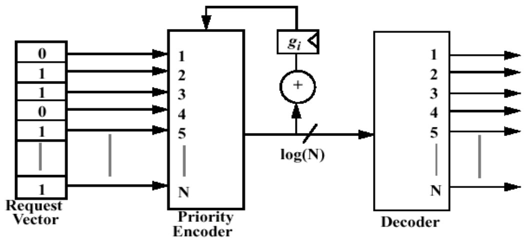


Figure 3.4: Implementation of Arbiter in iSLIP Scheduling Algorithm

Each iteration in an iSLIP scheduling algorithm consists of 3 steps:

- Step 1. *Request*: whenever a request arises at the input, it sends a request to every output for which it has a cell queued.
- Step 2. *Grant*: when request is accepted from the input, the output chooses the one that appears next in a fixed, round-robin schedule starting from the highest priority element. The input will be notified whether or not its request was granted.
- Step 3. *Accept*: when input request is granted, the one that appears next in a fixed, round-robin schedule starting from the highest priority element is accepted. The pointer is incremented by one location from the highest priority element of the round-robin schedule beyond the accepted output. When grant is accepted, pointers are updated after the first iteration. Pointers are updated only when grant is accepted otherwise pointers remain unchanged. Inputs and outputs which remained unmatched during earlier iterations are matched in next iteration.

In iSLIP scheduling algorithm lowest priority is given to the most recently made connection. For example if input ‘a’ successfully connects to output ‘b’, then both a and b are updated. The connection between a and b becomes the lowest priority connection in next cell time. Thus giving high throughput and fair chance for all inputs to make connections.
In iSLIP Scheduling, no connection will remain starving because an input will continue to request an output until it is successful. Pointers in iSLIP scheduling algorithm are not updated after the first iteration, an output will continue to grant highest priority

requesting input until it is successful. An iSLIP scheduler consists of 2N programmable priority encoders which is simple to implement. Under heavy load, all queues with a common output have the same throughput. The output pointer moves to each requesting input in a fixed order. Thus providing each with the same throughput.

3.3 Lonely Output Allocator:

Allocators such as single pass separable allocators choose the same popular output when many input arbiters request at same time without giving fair chance to few requests for some lonely outputs through the input stage.

Lonely output allocator overcomes this problem by adding a stage before the input arbiter which will count the number of requests for each output. Using this stage the input arbiter gives priority to those requests for outputs which have low request count. As a request vector is taken by first stage the number of requests competing for the requested output is determined. This gives a fair chance, a good matching and also reduces the number of conflicts at the output stage.

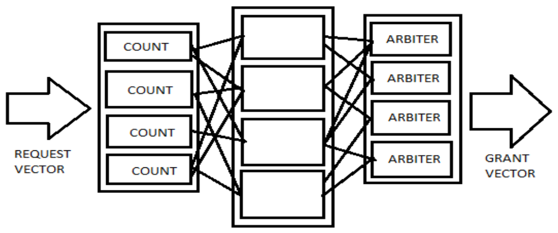


Figure 3.5: Block Diagram of Lonely Output Allocator

3.4 Wavefront Allocator:

Wavefront allocator arbitrates among requests for inputs and outputs simultaneously. It works by granting row and column tokens to a priority diagonal group. Consider the logic diagram of wavefront allocator in figure 3.6.

To grant a request arrived at the input, the cell needs to acquire both the column token ypri and row token xpri. The cell must be a member of the priority diagonal to generate row and column tokens. The tokens are passed along the xout and yout lines, when the cell that receives one or more tokens does not use them to grant a request.

These tokens propagate in a wavefront from the priority diagonal group so called as wavefront allocator. The propagation of tokens is stopped, if a cell with a request receives both a row and a column token because it was part of the original priority group because it was part of the original priority group or because the tokens were passed around the array.

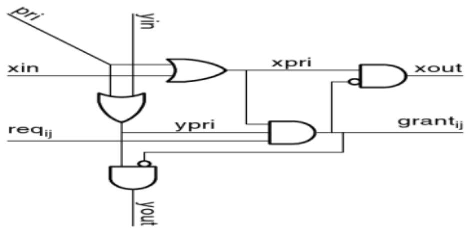


Figure 3.6: Wavefront Allocator Cell

The diagonal group reciveng the tokens is rotated at each cycle which improves fairness among all the cells. Consider an example in figure 3.7 where a 4X3 allocator with priority group 0, selected by signal p0 consists of cells Cij which includes X00, X13 and X22. All wavefront allocators must be square because each diagonal group must contain exactly one cell from each row and from each column.

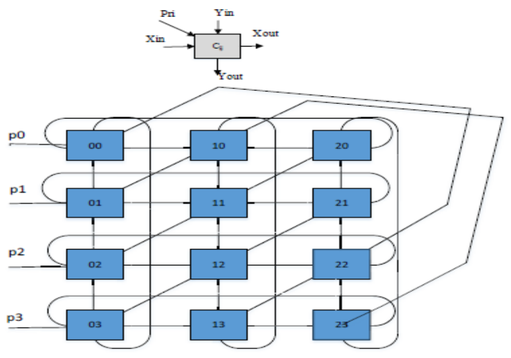


Figure 3.7: 4X3 Wavefront Allocator

IV. PROPOSED ALGORITHM

4.1 Flowchart of iSLIP Allocator

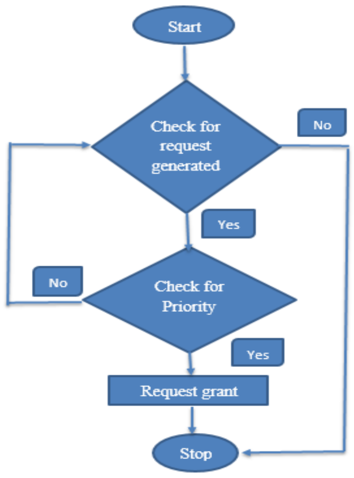


Figure 4.1: Flowchart of iSLIP Allocator

In iSLIP Allocator, when request is generated from inputs priority is checked. Request with highest priority is accepted to be granted. Once when highest priority request is completed, the process of accepting request is continued in a round robin fashion. The complete flow of iSLIP Allocator is shown in figure 4.1

4.2. Flowchart of Lonely Output Allocator:

Figure 4.2 shows flowchart of lonely output allocator. When requests are arrived, at each clock cycle request count for each request is checked. Depending upon this count value request is granted. After each clock cycle count value is incremented and gets updated depending upon the request generated. When all requests are served, the process is stopped.

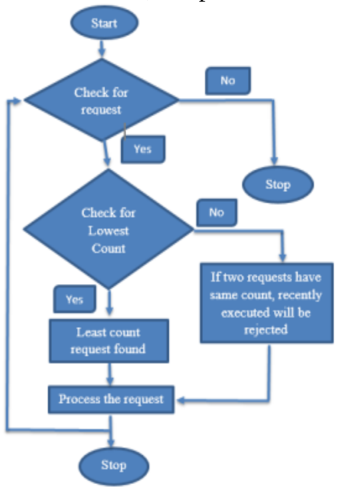

Start
Check for request
No
Yes
Stop
Check for Lowest Count
No
If two requests have same count, recently executed will be rejected
Yes
Least count request found
Process the request
Stop

Figure 4.2: Flowchart of Lonely Output Allocator

4.3 Flowchart of Wavefront Allocator:

Figure 4.3 shows the flowchart of Wavefront Allocator. The Wavefront Allocator requires the priority, row token and column token to process the request. When there is no priority generated to request or when there is no request to be granted then the token is passed to the next proceeding cell. It works only for a square matrix of cell. However non square matrix can add dummy rows and columns to form a square matrix.

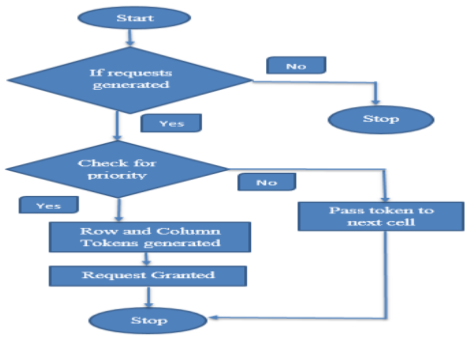
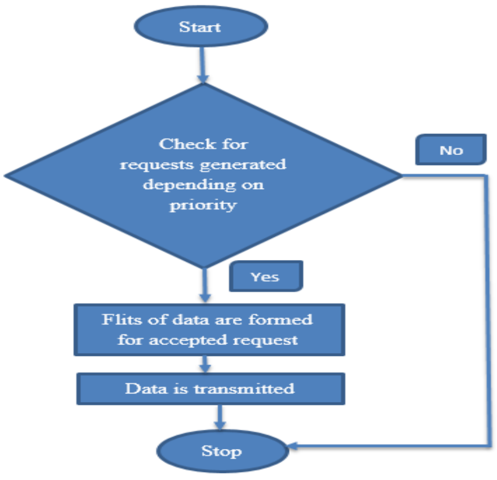


Figure 4.3: Flowchart of Wavefront Allocator

4.lowchart of Wormhole Allocator:

The Wormhole Allocator splits large data packet into smaller units called Flits. These flits are transmitted through buffers. They are stored in smaller size buffers and they use acknowledgement process to transmit data. Flits are transmitted when request is accepted. This process will avoid starvation.

If there is a large data to be transmitted then other requests need not wait for data to be transmitted. Fair and equal chance is given to all the requests. The flow of Wormhole Allocator is shown in the figure 4.4.


Start
Check for requests generated depending on priority
No
Yes
Flits of data are formed for accepted request
Data is transmitted
Stop
Figure 4.4: Flowchart of Wormhole Allocator

V. RESULTS:

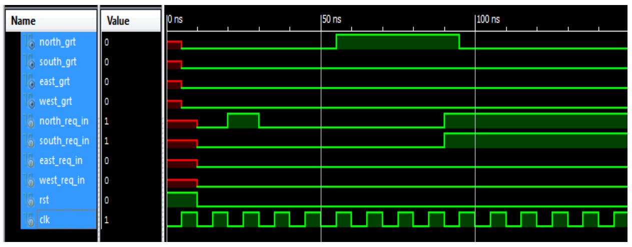


Figure 5. 1: Simulation results of iSLIP Allocator

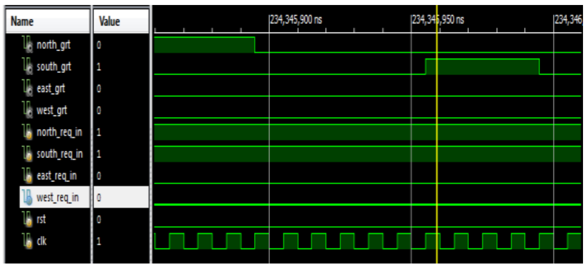


Figure 5.2: Simulation Results of Wavefront Allocator

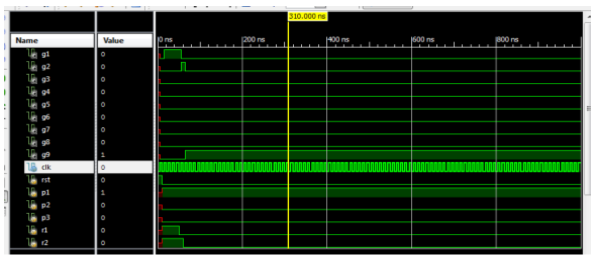


Figure 5.3: Simulation results of Wormhole Allocator

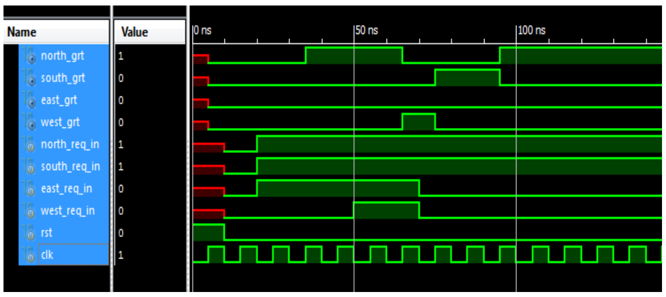


Figure 5.4: Simulation Results of Lonely Output Allocator

VII. CONCLUSION:

Synthesis and performance analysis of various scheduling algorithms is carried out in the project. The results showed different methods and ways of using available resources in FPGA device. Totally the tradeoff between throughput, area and power will determine correct approach of implementing Scheduling algorithm on on-chip router.

.

REFERENCES:

1.) Zhonghai Lu, "Using Wormhole Switching for Network on Chips: Feasibility Analysis and Microarchitecture Adaption," in Royal Institute of Technology, Department of Microelectronics and Information Technology, May 2005

2.) Jin Ouyang, Yuan Xie, "LOFT: A High Performance Network-on-Chip Providing Quality-of-Service Support," in 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO),pp 409-420, Dec 2010.

3.) Yuan-Ying Chang, Huang, "Y.S.-C., Poremba, M. Narayanan, V.Yuan, Xie King, C, "Title TS-Router: On maximizing the Quality-of- Allocation in the On-Chip Network," in IEEE 19th International Symposium on High Performance Computer Architecture (HPCA2013), pp 390-399, Feb 2013

4.) J. Guo, J. Yao, Laxmi Bhuyan, "An efficient packet scheduling algorithm in network processors," in proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies ,pp 807- 818, march 2005

5.) Akram Ben Ahmaed, Abderazek Ben Abdallah, Kenichi, Kuroda, "Architecture and Design of Efficient 3D Network-on-Chip (3D NoC) for custom multicore SoC," in International conference on Broadband, Wireless Computing, communication and Application, FIT, Fukuoka, Japan, Nov 2010

6.) G. jose, Delgado-Frias, B.R Girish, "A VLSI Crossbar switch with Wrapped Wave Front Arbitration," IEEE transaction on circuits and systems-I: Fundamental theory and applications, Vol 50, No 1, Jan 2003

7.) M. Karol and M. Hluchyj, "Queuing in high-performance packets switching," IEEE J. Select. Areas Common, vol. 6, pp. 1587–1597, Dec. 1988

8.) Nick McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches" IEEE/ACM transactions on Networking, vol 7, no,2, April 1999.

9.) P. Gupta, N. Mckeown, "Designing and implementing a Fast Crossbar Scheduler," in proc of Micro, IEEE, vol 19, no 1, pp. 20-28, Feb 1999.

10.) B. Osterloh, H. Michalik, B. Fiethe, K.Kotarowski, "SoCWire: A Network-on-Chip Approach for Reconfigurable System-on-Chip Designs in Space Applications," in proc of NASA/ESA Conference on Adaptive Hardware and Systems, pp 51-56, June 2008.

11.) Abdelrasul Maher, R. Mohhamed, G. Victor., "Evaluation of The Scalability of Round Robin Arbiters for NoC Routers on FPGA,"7th International symposium on Embedded Multicore/Many core System-on chip, pp61-66, 2013

12.) T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," ACM Trans. Comput. Syst., vol. 11, no. 4, pp. 319–352, Nov. 1993.

13.) B. Phanibhushana, K. Ganeshpure, S. Kundu, "Task model for on-chip communication infrastructure design for multicore systems," in proc of IEEE 29th International Conference on Computer Design (ICCD), pp 360-365, oct 2011.