Rate-based Scheduling Policy for QoS Flows in Networks on Chip

Aline Mello, Ney Calazans, Fernando Moraes
Faculdade de Informática - Pontifícia Universidade Católica do Rio Grande do Sul, PUCRS - Porto Alegre, Brazil
{alinev, calazans, moraes}@inf.pucrs.br

Abstract

Several propositions of NoC architectures claim providing quality of service (QoS) guarantees, which is essential for e.g. real time and multimedia applications. The state-of-art in NoC literature provides QoS at design time, using circuit switching and/or priority-based scheduling. Both methods optimize a given network template to achieve the QoS requirements after traffic generation and network simulation. However, modern SoCs may execute applications not devised at design time, and these may easily have its QoS requirements violated by a previously fixed NoC structure. This paper proposes a method to achieve QoS requirements in NoCs at execution time. The proposed rate-based scheduling policy is employed to determine the priority of each QoS flow being transmitted through the network. The basis of this scheduling method is the difference between the rate required by a given flow and the rate currently used by this flow. This difference corresponds to the flow *priority* used by the scheduler. Differently from traditional priority-based scheduling, the priority is dynamically adjusted. Preliminary results show the efficiency of the rate-based scheduling to meet QoS requirements, by comparing the proposed scheduling to priority-based scheduling.

1. Introduction

Network-on-Chip (NoC) is an emerging paradigm for communications within large VLSI systems implemented on a single silicon chip, known as System on Chip (SoC). In a NoC based SoC, modules such as processor cores, memories and other specialized IP blocks exchange data encoded in packets, using the NoC as a subsystem for data transport.

Distributed multimedia applications (e.g. 3G phones), need to communicate in real-time and are sensitive to the quality of services (QoS) they receive from the NoC. The term QoS refers here to the capacity to control the communication infra-structure to meet the application design requirements in what concerns the communication among modules of the SoC.

Usually, NoC literature employs two services class definitions: best effort (BES), and guaranteed (GS). BE services guarantee the transmission of all packets from a given source to a given target without any temporal bound guarantee. GS provide rigid bounds on one or on a subset of performance figures such as throughput (GT), latency, jitter and packet losses. This paper proposes to add a new service class, named Quality Services (QS). QS is defined as a service class where the network actively tries to reach application requirements without guaranteeing rigid bounds for performance figures. Three reasons can be advanced to propose this new service class. First, the typical workload of SoC applications is tolerant to limited variation in performance figures, possibly not requiring GS. Second, although GS can locally provide the best possible level of services, in general QS is capable of achieving a better level of global performance. Third, QS allows a better use of resources, leading naturally to a better dimensioning of the NoC.

Most current NoC implementations only provide support to BE services [1], including commercial products such as Arteris [2]. BE services are inadequate to satisfy QoS requirements for applications/modules with tight performance requirements.

NoC implementations providing support to QoS ([2]-[12]) try to achieve performance requirements at *design time*. This requires application traffic modeling, system simulation and NoC optimization and/or sizing. The internal router architecture of such NoCs employs circuit switching and/or priority-based scheduling to attain performance requirements for a given application. Circuit switching allows implementing GS and priority-based scheduling is a technique to meet QS.

Several modern SoCs may execute applications not devised at design time, and these may easily have its QoS requirements violated by a previously fixed NoC structure. Also, designing a NoC to support any traffic scenario is often unfeasible in terms of power and area.

The objective of this paper is to propose and evaluate a method to achieve QoS requirements at *execution time* for NoCs using QS. The proposed method uses a rate-based scheduling policy, being a two-step process. First, a data flow requiring QoS (called a *QoS flow*) is admitted in the NoC if and only if the NoC can transmit the rate required by the specific flow end-to-end, in what is called admission control. Next, each router dynamically defines the priority of each QoS flow locally, as a function of the rate used by this flow.

This paper is organized as follows. Section 2 presents related work in NoCs that offer support to obtain QoS, discussing limitation of current methods. Section 3 details the proposed scheduling method for QoS flows. Section 4 evaluates the proposed method, comparing it to priority-based scheduling method. Finally, Section 5 presents conclusions and directions for future work.

978-1-4244-1710-0/07/$25.00 © 2007 IEEE

2. Related Work

Current NoC designs employ at least one of three methods to provide QoS: (*i*) dimensioning the network to provide enough bandwidth to satisfy all IP requirements in the system; (*ii*) providing support to circuit switching for all or for selected IPs; (*iii*) making available priority-based scheduling for packet transmission.

Harmanci et al. [3] present a quantitative comparison between circuit switching and priority-based scheduling, showing that the prioritization of flows on top of a connectionless communication network is able to guarantee end-to-end delays in a more stable form than circuit switching. However, the reference does not quantify results numerically. A possible explanation for this is the use of a TLM SystemC modeling, instead of clock cycle accurate models advanced here. Additionally, the structural limitations of circuit switching and priority-based scheduling are not depicted.

The first method to provide QoS mentioned above is advocated e.g. by the Xpipes NoC [4]. A designer sizes Xpipes according to application requirements, adjusting each channel bandwidth to fulfill the requirements. However, applying this method alone does not guarantee avoidance of local congestions (hot spots), even if bandwidth is largely increased. This fact, coupled to ever-increasing performance requirements [5], render the method inadequate to satisfy requirements for a wide range of distinct applications.

The second method, support to circuit switching[1], provides a connection-oriented distinction between flows. This method is used in Æthereal [6], aSOC [7], Octagon [8], Nostrum [9] and SoCBUS [10] NoCs. For example, the Nostrum NoC [9] employs virtual circuits (VC), with the routing of QoS flows decided at design time. The communications on the physical channels are globally scheduled in time slots (TDM). The VCs guarantee throughput and constant latency at execution time, even with variable traffic rates. Circuit switching NoCs create connections for each or to selected flows. The establishment of connections requires allocation of resources such as buffers and/or channel bandwidth. This scheme has the advantage of guaranteeing tight temporal bounds for individual flows. However, this method has two main disadvantages: (*i*) poor scalability [3]; (*ii*) inefficient bandwidth usage. Here, router area grows proportional to the number of supported connections, penalizing scalability. Resource allocation for a given flow is based in worst case scenarios. Consequently, network resources may be wasted, particularly for bursty flows.

QNoC [11], DiffServ-NoC [3] and RSoC [12] are examples of NoCs adopting the third method, packet switching with priorities. This connectionless technique groups traffic flows into different classes, with different service levels for each class. The method requires separate buffering to manipulate packets according to the services levels. To each service level corresponds a priority class. The network always serves first non-empty higher priority buffers. Packets stored in lower priority buffers are transmitted only when there are no higher priority packets waiting to be served. This scheme offers better adaptation to varying network traffic and a potentially better utilization of network resources. However, end-to-end latency and throughput cannot be guaranteed, except to the higher priority flows. Also, it is necessary to devise some form of starvation prevention for lower priority flows. When flows share resources, even higher priority flows can have an unpredictable behavior. Consequently, this method often provides a poorer QoS support than circuit switching.

Neither circuit switching nor priority methods guarantee QoS for *concurrent multiple flows*. When using the circuit switching method, the network may reject a number of flows, due to limited amount of simultaneously supported connections, even if network bandwidth is available. When multiple flows with the same priority compete for the same resources, priority-based networks have behavior similar to BE service networks [13]. As mentioned before, networks using any of the three above described methods employ techniques at design time to guarantee QoS, through traffic modeling, simulation-based network sizing (topology, buffer depth, flit width) and network synthesis. The drawbacks of sizing the network at design time are: (*i*) the complexity of traffic modeling and system simulation is very high, being thus error-prone; and (*ii*) the network designed in this way may not guarantee QoS for new applications. The first drawback may force the use of simplified application/environment models, which can in turn lead to incorrect dimensioning of the NoC parameters for synthesis. The second drawback may arise if new applications must run on the system after some initial implementation, as occurs with reconfigurable or programmable systems.

The main performance figures used in the above reviewed NoCs are end-the-end latency and throughput. Nonetheless, when QoS is considered, another concept can be of relevance, *jitter*. Jitter can be defined as the variation in latency, caused by network congestion, or route variations [14]. In connectionless networks, buffers introduce jitter. When packets are blocked, latency increases. Once the network can release packets from blocking, latency reduces, due to burst packet diffusion. Therefore, networks using only priorities cannot guarantee controlled jitter.

Some other works advocate different methods to enhance QoS. For example, Andreasson and Kumar proposed a *slack-time aware* routing [15][16], a source routing technique to improve overall network utilization by dynamically controlling the injection of BE packets in the network in specific paths, while guaranteed throughput (GT) packets are not employing those paths. This work is not directly related to QoS achievement.

[1] In this paper, the term *circuit switching* is used to refer to both, networks providing physical level structures to establish connection between source and destination, as well as to packet switched networks that employ higher level services (such as virtual circuits) to establish connections.

3. Rate-based Scheduling Policy

The proposed scheduling policy assumes the following NoC features: wormhole packet switching, deterministic routing, and physical channels multiplexed in at least two virtual channels (VC). BE flows are transmitted using only one VC, while QoS flows may use any VC. This resource reservation for QoS flows is necessary to avoid that multiple BE flows block the possibility of using the channel for some QoS flow. The proposed policy is a two-step process: admission control followed by dynamic scheduling.

The admission step determines if the network may accept a new QoS flow without penalizing performance guarantees already assigned to other QoS flows. The admission step starts by sending a control packet from the source router to the target router, containing the rate required by the IP. The QoS flow is admitted into the network if and only if all routers in the path to the target can transmit at the required rate. When the control packet arrives at the target, an acknowledgment signal is back propagated to the source router. This process is similar to the connection establishment in circuit switching, but differently from circuit switching there is no static resource reservation.

When the QoS flow is admitted, a *virtual connection* is established between the source and target router, as in ATM [17] networks. This virtual connection corresponds to a line in the *flow* table (see Figure 1) of each router in the connection path. Each line of the flow table identifies the QoS flow using the following fields: source router, target router, required rate, and used rate. The *flow table* depth determines how many simultaneous QoS flows can be admitted by each router. The virtual connection is released by the source router with another control packet.

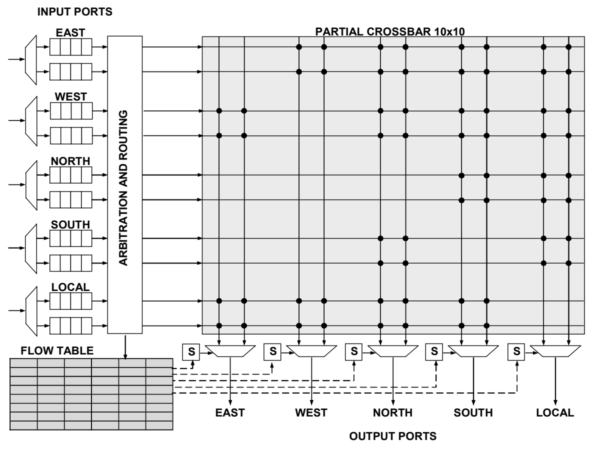


Figure 1. Router architecture with support for rate-based scheduling.

Once the virtual path is established, the source router may start sending QoS flow packets. When packets arrive at a router input port they are stored in input buffers, arbitrated (e.g. using round robin) and routed (e.g. XY deterministic algorithm) to an output port (Figure 1). Packets assigned to the same output port are served according to the proposed scheduling policy.

The implemented scheduling policy adopts a work-conserving mechanism, which is idle only when there is no packet awaiting to be served. BE flows are transmitted only when no QoS flows require the physical channel. When two or more Qos flows compete, the higher priority flow is scheduled first.

As illustrated in Figure 1, the flow table is read by the scheduler (blocks named S in Figure 1) to find the priority of each QoS flow assigned to a same output port. The QoS flow *priority* is the difference between the required rate and the rate currently used by the QoS flow. The flow priority is periodically updated according to Equation 1. A positive priority means that the flow used less rate than required in the considered sampling period. A negative priority means that the flow is violating its admitted rate in the considered sampling period.

$$priority_i = required\ rate - used\ rate_i \qquad (1)$$

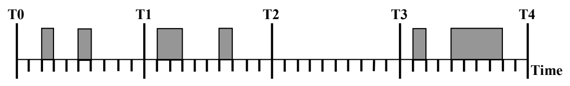
The required rate is fixed during the admission control step. The used rate (UR) is *periodically* computed according to Equation 2:

$$UR_i = \begin{cases} CR_i, & if\ UR_{i-1} = 0 \\ \dfrac{UR_{i-1} + CR_i}{2}, & if\ UR_{i-1} \neq 0 \end{cases} \qquad (2)$$

where:

- *CR*: is the *current rate* used during the current period;
- *UR*: is the average of the previous used rate and the current used rate.

Figure 2 illustrates packets of a given QoS flow being transmitted. Timestamps T0 to T4 designate when the rates are sampled, assuming in this example 10 time units in each interval. The table in the Figure corresponds to the behavior of one flow table from T0 to T4.



Time	T0	T1	T2	T3	T4
Source	01	01	01	01	01
Target	55	55	55	55	55
Required rate	25%	25%	25%	25%	25%
Current rate (**CR**)	0%	20%	30%	0%	50%
Used rate (**UR**)	0%	20%	25%	12%	31%
Actual rate	0%	20%	25%	16%	25%
Priority	**25**	**5**	**0**	**13**	**-6**

Figure 2. Transmission of packets for a given QoS flow.

In this example, the 4th line of the table contains the required rate (25%) for this flow. At timestamp T1 the current rate (5th line) is 20%, corresponding to the channel bandwidth used by the flow in the previous interval (T0-T1). According to the Equation 2 it is possible to obtain the used rate (6th line of the table). The 8th line of the table contains the flow priority, which is updated according to Equation 1.

The interval between timestamps is an important parameter of the proposed method. The 7th line contains the actual flow rate (shown here for comparison purposes, not present in the flow table). If the chosen interval is too short, the computed used rate may not correspond to the actual rate, compromising the scheduling method. If the interval is too long, the computed used rate will be accurate, but the flow priority will remain fixed for a long period, also compromising the method.

To minimize the error induced by the sampling period, the method in fact employs two sample intervals. In the previously presented example, consider a second current rate (*CR2*) and a sample interval 4 times larger than the original one. In this example, *CR2* will be equal to 100% (summation of *CR* from T0 to T4) in T4. Dividing *CR2* by 4, the corrected used rate is obtained (*CUR*, Equation 3). It can be observed that applying *CUR* to *UR* each *n* intervals (4 in this example), the error is minimized.

$$CUR = \frac{CR2}{n} = \frac{\sum_{i=0}^{n-1} CR_i}{n} \qquad (3)$$

Consequently, in Equation 1 UR_i receives *CUR* when *i mod n* is equal to zero, where *n* corresponds to the result of dividing the long sample interval value by the short sample interval value.

It is important to mention that if only the used rate were considered in the priority computation ($priority_i = 100 - UR_i$), the scheduling policy tends to balance physical channel use. This implies disregarding the fact that distinct QoS flows may require distinct rates. Consider two QoS flows with instantaneous UR of 20% and 30%, respectively. The first flow would be scheduled first. Assume that the required rates are 10% and 40% to each flow. The first flow scheduled first does not consider that it is using more bandwidth than the required. Using Equation 1 the correct schedule is obtained.

4. Experimental Results

This Section compares the performance of the priority-based scheduling with the proposed rate-based scheduling, since both support QS. Traffic injection and results capture is modeled with SystemC, while the NoC is modeled through RTL VHDL [18]. The NoC parameters are: 8x8 mesh topology; XY routing; 16-bit flits; 2 virtual channels; 8-flit buffers associated to each virtual channel.

4.1 Experimental Setup

Table 1 presents the flows used in the experiments. Flow A is characterized as a CBR service with QoS requirements, as latency and jitter. Nodes generating flows A transmit 2000 packets. The results do not take into account the first 50 packets and the last 50 packets. They are discarded from results, since the traffic at the beginning and the end of the simulation does not correspond to regular load operation. Flow B is a BE flow modeled using a Pareto distribution. This flow is used to disturb QoS flows, being considered as noise traffic. For this reason, results for the B flow are not discussed.

Table 1. Flows Characterization.

Type	Service	QoS	Distribution	Number of Packets	Packet Size	Target
A	CBR	Yes	Uniform (20%/30%)	2000	50, 100, 200, 500	Fixed
B	BE	No	Pareto (20% on)	Random	20	Random

Figure 3 presents the spatial distribution of source and target nodes. In this scenario, two QoS flows (F1 and F2) originated at different nodes share part of the paths to targets. The remaining network nodes transmit B flows, disturbing the QoS flows.

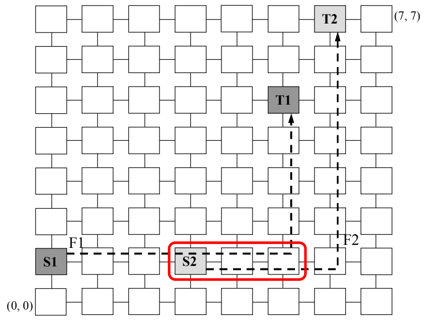


Figure 3. Spatial distribution of source and target nodes for flows with QoS requirements. Dotted lines indicate the path of each flow. Rounded rectangles highlight the area where flows compete for network resources.

Equation 4 gives the ideal latency to transfer a packet from a source to a target, in clock cycles.

$$ideal\ latency = 5N + P \qquad (4)$$

where:

- 5: is the router minimal latency (arbitration and routing);
- *N*: is the number of routers in the communication path (source and target included);
- *P*: is the packet size.

When the packet size is 50 flits, the ideal latency for the scenario presented in the Figure 3 is 100 (5x10+50) clock cycles for both scheduling.

4.2 Results

Table 2 presents the latency values, jitter and throughput when the packet size is 50 flits and the inserted rate is 20%. Both scheduling policies guarantee throughput close to the inserted rate (20%).

Analyzing the priority-based scheduling, F2 has average latency near to ideal, while F1 flow has higher latency (average latency 44% far from the ideal latency). F1 and F2 are CBR flows with the same priority, competing for the same resources. They insert packets in the network at fixed intervals. As the F2 source node is closer to the region disputed by the flows, it is always served first. This experiment demonstrates that priority-based scheduling is inefficient for QS when flows with the same priority compete for the same resources.

Table 2. Results for flows F1 and F2, 50 flits, 20% load.

Performance Figures		Priority-based		Rate-based	
		F1	F2	F1	F2
Latency	**Ideal (ck)**	**100,00**	**100,00**	**100,00**	**100,00**
	Minimum (ck)	141,00	100,00	119,00	119,00
	Average (ck)	**144,23**	**101,78**	**148,95**	**121,93**
	Maximal (ck)	154,00	133,00	174,00	133,00
Jitter (ck)		2,66	3,04	18,63	3,03
Average throughput (%)		19,21	19,21	19,35	19,20

In the rate-based scheduling, the priority is dynamically updated according to the used rate, not as a function of the arrival time of the packets in the router. Therefore, as both flows have the same required rate, the bandwidth is equally divided between the flows, reducing the difference between the F1 and F2 average latency from 42% (when priority-based scheduling is used) to 22%. The jitter is increased when compared to priority-based scheduling because F1 and F2 are not served always in the same order.

Table 3 displays the latency values, jitter and throughput when the packet size is 50 flits and the inserted rate is 30%. As presented in Table 2, both scheduling policies guarantee throughput close to the inserted rate (30%).

Table 3. Results for flows F1 and F2, 50 flits, 30% load.

Performance Figures		Priority-based		Rate-based	
		F1	F2	F1	F2
Latency	**Ideal (ck)**	**100,00**	**100,00**	**100,00**	**100,00**
	Minimum (ck)	141,00	100,00	119,00	119,00
	Average (ck)	**143,82**	**101,44**	**137,31**	**121,94**
	Maximal (ck)	156,00	112,00	184,00	131,00
Jitter (ck)		2,75	2,77	21,20	2,47
Average throughput (%)		28,80	28,81	29,60	28,80

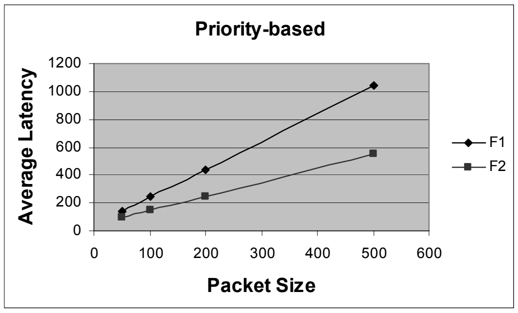
It is possible to observe that average latency and jitter of QoS flows (F1 and F2) presented in Table 3 have similar behavior of Table 2. The reasons for the similar behavior are: (1) F1 and F2 are CBR flows with the same inserted rate (fixed intervals between packets); (2) the total used load by these flows is inferior to 100%, allowing them to be transmitted with the same delay; (3) F1 and F2 are priority flows that only compete between themselves for the same resources. In priority-based scheduling, the disturbing traffic does not interfere the QoS flows due to its lower priority. However, in rate-based scheduling, F1 average latency is slightly reduced when the injection rate has increased. The reason is the amount of conflicts between BE and QoS flows in the shared virtual channel[2], which changes with the injection rate.

[2] One virtual channel is reserved for QoS flows, while the second one is shared between QoS and BE flows. QoS flows may use the shared virtual channel when no BE packet is being transmitted.

Figure 4 presents the F1 and F2 average latency when the packet size is 50, 100, 200, and 500 flits. Figure 4(a) shows the behavior of the priority-based scheduling. In this Figure it is possible to observe the difference between the F1 and F2 average latency increases according to the packet size. As mentioned before, the F2 average latency is smaller because F2 source node is closer to the region disputed by the flows. Analyzing the rate-based scheduling presented in Figure 4(b), the F2 average latency is slightly higher when compared to the priority-based scheduling. However, the difference between the F1 and F2 average latency is significantly reduced.

This results shows the superiority of the rate-based scheduling over the priority-based scheduling, allowing to deliver QoS packets with similar latency, independently of the packet size.

(a)



(b)

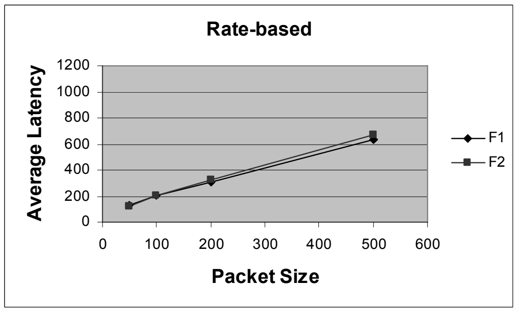


Figure 4. Average latency for F1 and F2 flows, Experiment I, CBR Traffic. (a) priority-based scheduling; (b) rate-based scheduling.

Sample periods are a critical factor in the rate-based method, since they define how frequently priorities are updated. Rate-based results, presented in Table 2 and Table 3, employed short and long sample periods equal to 1000 and 8000 clock cycles, respectively. Small differences (less than 1%) were observed when samples periods were reduced to 250/2000 clock cycles and packet size equals to 50 flits. However, reducing the samples periods to 250/2000 clock cycles and packet size equals to 200 flits increased the overall latency. These results show that a trade-off between packet size and samples period should be established, as discussed in Section 3.

5. CONCLUSIONS AND FUTURE WORK

As discussed, the state of the art in NoCs still does not provide efficient solutions to achieve QoS for applications when the network traffic is not known in advance. The main drawback of circuit switching and priority-based scheduling is the performance unpredictability when QoS flows compete for the same network resources. This paper presented a rate-based scheduling policy, which adjusts the flow priority w.r.t. the required flow rate and current rate used by the flow.

Good results were obtained with CBR flows. When QoS flows with the same priority compete for resources, priority-based scheduling favors the flow that reaches the shared resources first, penalizing the latency of the second arriving flow. Rate-based scheduling overcomes this problem, balancing flows according to their required rates.

As future work it is possible to enumerate: (*i*) evaluating the proposed method with more experiments; (*ii*) evaluating area overhead of the approach, which is expected to be small, because only a small table and few counters were added to the NoC router; (*iii*) implementing congestion control mechanisms; (*iv*) developing services in superior layers to the network layer, allowing to include the requirements specification and to verify if the network is supporting these requirements.

6. ACKNOWLEDGMENTS

This research was supported partially by CNPq (Brazilian Research Agency), project 300774/2006-0.

7. REFERENCES

[1] Di Micheli, G.; Benini, L. "*Networks on Chips: Technology and Tools*". Morgan Kaufmann, 2006, 304 p.

[2] Arteris. "*Arteris Network on Chip Company*". 2005. Available at http://www.arteris.net.

[3] Harmanci, M.D.; Escudero, N.P.; Leblebici, Y.; Ienne, P. "*Quantitative Modelling and Comparison of Communication Schemes to Guarantee Quality-of-Service in Networks-on-Chip*". In: ISCAS, 2005, pp. 1782-1785.

[4] Bertozzi, D.; Benini, L. "*Xpipes: A Network-on-chip Architecture for Gigascale Systems-on-Chip*". IEEE Circuits and Systems Magazine, v.4(2), 2004, pp. 18-31.

[5] Shin, J.; Lee, D.; Kuo, C.-C. "*Quality of Service for Internet Multimedia*". Prentice Hall, 2003, 204 p.

[6] Goossens, K.; Dielissen, J.; Radulescu, A. "*Æthereal Network on Chip: Concepts, Architectures, and Implementations*". IEEE Design and Test of Computers, v.22(5), 2005, pp. 414-421.

[7] Liang, J.; Swaminathan, S.; Tessier, R. "*aSOC: A Scalable, Single-Chip communications Architecture*". In: IEEE International Conference on Parallel Architectures and Compilation Techniques, 2000, pp. 37-46.

[8] Karim, F.; Nguyen, A.; Dey S. "*An interconnect architecture for network systems on chips*". IEEE Micro, v.22(5), 2002, pp. 36-45.

[9] Millberg, M.; Nilsson, E.; Thid, R.; Jantsch, A. "*Guaranteed Bandwidth Using Looped Containers in Temporally Disjoint Networks Within the NOSTRUM Network on Chip*". In: DATE, 2004, pp. 890-895.

[10] Wiklund, D.; Liu D. "*SoCBUS: Switched Network on Chip for Hard Real Time Systems*". In: IPDPS, 2003, 8p.

[11] Bolotin, E; Cidon, I.; Ginosar R.; Kolodny A. "*QNoC: QoS Architecture and Design Process for Network on Chip*". Journal of Systems Architecture, v.50(2-3), 2004, pp 105-128.

[12] Véstias, M.; Neto, H. "*A Reconfigurable SoC Platform Based on a Network on Chip Architecture with QoS*". In: XX DCIS, 2005 , 6 p.

[13] Mello, A.; Tedesco, L.; Calazans, N.; Moraes, F. "*Evaluation of Current QoS Mechanisms in Networks on Chip*". In: International Symposium on System-on-Chip, v.1, 2006, pp. 115-118.

[14] Dally, W.J.; Towles, B. "*Principles and Practices of Interconnection Networks*". Morgan Kaufmann Publishers, 2004, 550p.

[15] Andreasson, D.; Kumar, S. "*Improving BE Traffic QoS Using GT Slack in NoC Systems*". In: NORCHIP, 2005, pp. 44-47.

[16] Kumar, S.; Andreasson, D. "*Slack-Time Aware Routing in NoC Systems*". In: ISCAS, 2005, pp. 2353-2356.

[17] Giroux, N.; Ganti, S. "*Quality of Service in ATM Networks: State-of-Art Traffic Management*". Prentice Hall, 1998, 252p.

[18] Moraes, F.; Calazans, N.; Mello, A.; Möller, L.; Ost, L. "*Hermes: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip*". Integration the VLSI Journal, v.38(1), Oct. 2004, pp. 69-93.