QoS in Networks-on-Chip - Beyond Priority and Circuit Switching Techniques

Aline Mello, Ney Calazans, Fernando Moraes

Pontifícia Universidade Católica do Rio Grande do Sul (FACIN-PUCRS)
Av. Ipiranga, 6681 - Prédio 30 / BLOCO 4 - 90619-900 - Porto Alegre – RS – BRASIL
{alinev, calazans, moraes}@inf.pucrs.br

Abstract. The idea behind the proposition of Networks-on-Chip (NoCs) for modern and future systems on chip capitalizes on the fact that busses do not scale well when shared by a large number of cores. Even if NoC research is a relatively young field, the literature abounds with propositions of NoC architectures. Several of these propositions claim providing quality of service (QoS) guarantees, which is essential for real time and multimedia applications. The most widespread approach to attain some degree of QoS guarantee relies on a two-step process. The first step is to characterize application performance through traffic modeling and simulation. The second step consists in tuning a given network template to achieve some degree of QoS guarantee. These QoS targeted NoC templates usually provide specialized structures to allow either the creation of connections (circuit switching) or the assignment of priorities to connectionless flows. It is possible to identify three drawbacks in this two-step process approach. First, it is not possible to guarantee QoS for new applications expected to run on the system, if those are defined after the network design phase. Second, even with end-to-end delay guarantees, connectionless approaches may introduce jitter. Third, to model traffic precisely for a complex application is a very hard task. If this problem is tackled by oversimplifying the modeling phase, errors may arise, leading to NoC parameterization that is poorly adapted to achieve the required QoS. This Chapter has two main objectives. The first one is to evaluate the area-performance trade-off and the limitations of circuit switching and priority scheduling to meet QoS. This evaluation will show where such implementations are really suited for QoS, and when more elaborate mechanisms to meet QoS are needed. The second objective comprises proposing a method, called *rate-based scheduling*, to approach QoS requirements considering the execution time state of the NoC. The evaluation of circuit switching and priority scheduling show that: (*i*) circuit switching can guarantee QoS only to a small number of flows; the technique do not scale well, and can potentially waste significant bandwidth; (*ii*) priority-based approaches may display best-effort behavior and, in worst-case situations, may lead to unacceptable latency for low priority flows, besides being subject to jitter. In face of these limitations, rate-based scheduling arises as an option to improve the performance of QoS flows when varying traffic scenarios are used.

Key words: quality of service, QoS, network-on-chip, NoC, circuit switching, priority scheduling, rate-based scheduling.

1 Introduction

As described in [1], networks on-chip (NoCs) are a promising way to implement future interconnection architectures, due to their: (*i*) energy efficiency and reliability [2]; (*ii*) scalability of bandwidth when compared to bus architectures; (*iii*) reusability; (iv) distributed routing decisions [2]. Network interfaces, routers and point-to-point links define a NoC infrastructure. A network interface connects IPs to the NoC, and is responsible to prepare and deliver packets or entire messages to other IPs through the NoC and to receive packets/messages from the network to the IP [2].

Currently, most NoC implementations only provide support to best effort (BE) services [1], even those proposed by NoC companies like Arteris [3]. BE services guarantee delivery of all packets from a source to a target, but provide no bounds for throughput, jitter or latency. This kind of service usually assigns the same priority to all packets, leading to unpredictable transmission delays. The term Quality of Service (QoS) refers to the capacity of a network to control traffic constraints to meet design requirements of an application or of some of its specific modules. Thus, BE services are inadequate to satisfy QoS requirements for applications/modules with tight performance requirements, as in the case of multimedia streams. To meet performance requirements and thus guarantee QoS, the network needs to include specific characteristics at some level in its protocol stack. Accessing the relative priority and requirements of each flow enables an efficient assignment of resources to flows [4].

Present NoC implementations providing support to QoS try to achieve performance requirements at *design time*. The network is designed according to the application, requiring accurate traffic modeling and simulation to obtain the desired bandwidth and latency figures for the target application. The simulation results allow dimensioning the network to support application requirements. Network synthesis occurs after simulation. However, it is still possible that QoS guarantee is not met for new applications. Modern SoCs, such as 3G phones, support different application profiles. Designing the network to support all possible traffic scenarios is unfeasible in terms of power and area. Thus, some mechanism has to be used at *execution time* to enable meeting QoS requirements for a wide range of applications. Some examples of mechanisms are those long used in IP and ATM networks, including admission control and traffic shaping. The main advantage of using such mechanisms is to support new applications after network design, at the cost of extra area and power.

NoCs proposed to meet QoS (e. g. [5] [6] [7] [8] [9] [10] [11] [12]) employ circuit switching and/or priority scheduling in their router architectures to attain performance requirements. Nonetheless, these techniques are implemented at *design time* for some devised traffic scenarios. For real applications, there can be a significant uncertainty during execution time: some flows may disappear and re-appear randomly or periodically, and they may also be interdependent [13]. To the knowledge of the Authors, there is no NoC implementation with built-in mechanisms to meet QoS taking into account the state of the network at *execution time*.

This Chapter has two objectives. The first is to evaluate area-performance trade-off and limitations of circuit switching and priority scheduling to meet QoS. This shows where such implementations are really suited for achieving QoS, and where more elaborate mechanisms are needed. The second is to propose a method, *rate-based scheduling*, to approach QoS requirements considering the NoC *execution time* state.

The rest of this Chapter is organized as follows. Section 2 is an overview of NoCs that offer guarantees of QoS. Section 3 details four NoC designs: (*i*) a best effort NoC; (*ii*) a static priority scheduling NoC; (*iii*) a NoC employing dynamic priority scheduling to meet QoS; and (*iv*) a NoC employing circuit and packet switching. Section 4 proposes the rate-based scheduling policy. Section 5 analyzes latency, jitter and throughput for all NoCs. Section 6 gives conclusions and suggests future works.

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; (*ii*) providing support to circuit switching for all or selected IPs; (*iii*) making available priority scheduling for packet transmission.

Harmanci et al. [14] present a quantitative comparison between circuit switching and priority 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. A possible explanation for this is the use of a TLM SystemC modeling, instead of clock cycle accurate models. Also, structural limitations of circuit switching and priority scheduling are not depicted.

The first method to provide QoS mentioned above is advocated e. g. by the Xpipes NoC [6]. The 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 [15], makes the method improper to satisfy a wide range of applications.

The second method, support to circuit switching[1], provides a connection-oriented distinction between flows. This method is used in Æthereal [7], aSOC [8], Octagon [9], Nostrum [10] and SoCBUS [11] NoCs. For example, the Nostrum NoC [10] 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 using TDM. The VCs guarantee throughput and constant latency at execution time, even with variable traffic rates. Circuit switching NoCs create connections for all 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, the method has two main disadvantages: (*i*) poor scalability [14]; (*ii*) inefficient bandwidth usage. The router area is proportional to the number of supported connections, penalizing

[1] Here, 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 employing higher level services (such as virtual circuits) to create connections.

scalability. Resource allocation for a given flow is based on worst case scenarios. Consequently, network resources may be wasted, particularly for bursty flows.

QNoC [5], DiffServ-NoC [14] 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. It requires separate buffering to manipulate packets according to the services levels. To each service level corresponds a priority class. The network serves non-empty higher priority buffers first. Packets stored in lower priority buffers are transmitted only when no higher priority packets is 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 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. Thus, this method often provides a poorer QoS support than circuit switching.

Neither circuit switching nor priority methods guarantee QoS for *multiple concurrent flows*. When using circuit switching, the network may reject flows, due to a limited amount of simultaneously supported connections, even if bandwidth is available. When multiple flows with the same priority compete for resources, priority-based networks have behavior similar to BE networks (see Section 5). As mentioned before, networks using any of the three methods above employ techniques at design time to guarantee QoS through traffic modeling, simulation-based network sizing 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 execute after product delivery, as occurs in reconfigurable or systems.

The main performance figures used in the above reviewed NoCs are end-the-end latency and throughput. But when QoS is considered, another concept can be of relevance, *jitter*, the variation in latency, caused by network congestion, or route variations [16]. 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. Thus, networks using only priorities cannot guarantee jitter control. Some works advocate different methods to enhance QoS. For example, Andreasson and Kumar proposed a *slack-time aware* routing [13] [17], a source routing technique to improve overall network utilization by dynamically controlling the injection of BE packets in the network at specific paths, while guaranteed throughput (GT) packets are not employing these. However, this work does not aim at QoS achievement.

The NoCs to be described in the next two Sections share a basic set of common features: 2D mesh topology, wormhole packet switching, deterministic distributed routing, and physical channels multiplexed in at least two virtual channels (VC). This certainly does not cover all possible features found in NoC architectures proposed in the literature. But many of these features are found in several NoCs [18] [19].

3 Reference NoC Designs

This Section presents four NoC designs. The first is a NoC supporting BE services only (BE-NoC). The second and third add priority schemes to the BE-NoC, to enable differentiating flows. The fourth design adds circuit-switching to the BE-NoC.

3.1 Best Effort NoC - *BE-NoC*

The BE-NoC is based on Hermes [19], a parameterizable infrastructure used to implement low area overhead packet switching NoCs with 2D mesh or torus topology, which allows to select the routing algorithm, the flit size and the buffer depth. This work employs Hermes with a parameterizable number of virtual channels (VCs) [20]. The first and the second flits of a packet are header information, respectively containing the target address, and the payload size (up to $2^{(\text{flit size, in bits})}$) in flits. The remainder flits are payload.

Credit based is the flow control algorithm assumed here. **Fig. 1** shows the credit-based interface between routers. The output port signals are: (*1*) clock_tx: synchronizes data transmission; (*2*) tx: indicates data availability; (*3*) lane_tx: indicates the VC or lane transmitting data; (*4*) data_out: data to be sent; (*5*) credit_in: indicates available buffer space, for each lane.

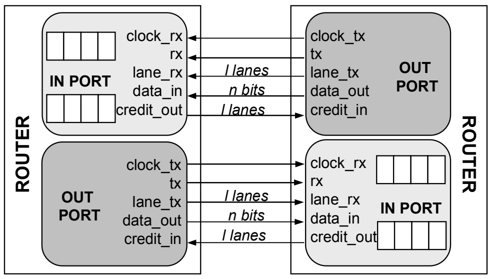


Fig. 1. Physical router interface for the BE-NoC.

The router has centralized switching control logic and five bi-directional ports: East, West, North, South, and Local. The Local port establishes a communication between the router and its local core. The other ports connect to neighbor routers. Any physical channel may support multiplexed VCs. **Fig. 2** presents the internal router structure, with two lanes per physical channel. Although physical channel multiplexing may increase switching performance [1], it is important to keep a compromise among performance, complexity and router area.

Each input port has a depth *d* buffer for temporary flit storage. When *n* lanes are used, a buffer with *d/n* depth is associated to each lane. The input port receives flits, storing them in the buffer indicated by signal *lane_rx* (**Fig. 1**). Next, it decrements the amount of lane credits. When an output port transmits a flit, this flit is removed from the buffer and the credit counter is incremented. Credit availability reaches a neighbor router through signal credit_out (**Fig. 1**).

The XY routing algorithm sends packets in the X direction up to the target X coordinate, and then proceeds in the Y direction until reaching the target router. The behavior of this algorithm allows the use of the partial crossbar of **Fig. 2**. Packets coming from the Local, East or West ports may go to any output port. Packets coming from the North port can only be transmitted to the South and Local ports, and packets coming from the South port can only be follow to the North and Local ports. A partial crossbar reduces router area by up to 3%, compared to a full crossbar.

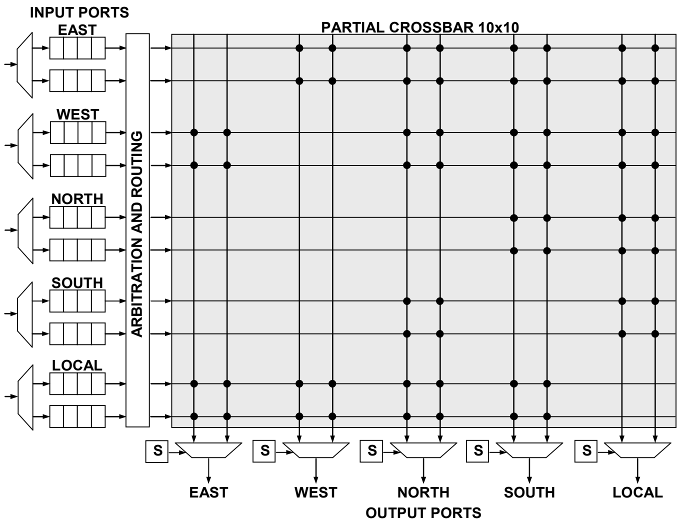


Fig. 2. Router internal structure, for two virtual channels Hermes NoC. “Solder points” indicate existing connections in the partial crossbar.

Multiple packets may arrive simultaneously in a given router. A centralized round-robin arbitration grants access to incoming packets. The priority of a lane is a function of the last lane having a routing request granted. If the incoming packet request is granted by the arbiter, the XY routing algorithm is executed to connect the input port to the correct output port. When the algorithm returns a busy output port, the header flit and all subsequent flits of this packet are blocked.

After routing execution, the output port allocates the bandwidth among the *n* lanes. Each lane with flits to transmit occupies at least *1/n* of the physical channel bandwidth. If only one lane satisfies this condition, it occupies the whole physical channel bandwidth. After all flits in a packet are transmitted, the port is released.

3.2 Static Priority NoC – *SP-NoC*

The objective of this design is to add the ability to provide differentiated services to the flows, using a resource allocation mechanism based on static priorities (similar to QNoC [5]). This design modifies the arbitration and scheduling router policies without modifying the BE-NoC router interface.

In the *SP-NoC*, each lane is associated to a priority and is served according to it. The priority of each lane is given by its index, as defined by Equation 1. In this way, this NoC allows the network to differentiate *n* flows, where *n* is the number of lanes per physical channel.

$$\textit{priority of } L_i = i - 1, \text{ for all } i \geq 1 \qquad \textbf{(1)}$$

To differentiate flows, the packet header is extended by a new field, named *priority*. This field determines which lane is used for packet transmission. For example, lane L2 transmits packets with priority 1. The user may assign to the priority field a value between *0* (lowest priority) and *n-1* (highest priority) . Only the source router verifies the priority field. The remaining routers transmit packets using the same lane allocated by the source router.

The assignment of priorities to virtual channels requires modification of router arbitration and scheduling algorithms. In priority-based arbitration, when multiple packets arrive simultaneously at the router input ports, the packet with higher priority is served first, even if other packets are waiting to be served. In priority-based scheduling[2], packets with higher priority are also served first. Then, data transmission in lower priority lanes depends on the load of the higher priority lanes, which can vary dynamically. For this reason, end-to-end latency bounds cannot be determined for all lanes, only for the highest priority lane. Consequently, it is hard to support multiple services with guaranteed QoS using priorities [21].

Priority-based arbitration and scheduling are effective for a small number of virtual channels [21]. For example, it is possible to reserve a virtual channel for real-time flows, a second one for non-real-time flows with controlled losses, and a third one for best-effort traffic. The drawback of the approach is the fact that the router area increases approximately with the square of the number of virtual channels [20].

3.3 Dynamic Priority NoC – *DP-NoC*

In a *DP-NoC*, priority is assigned to *flows* as opposed to the *SP-NoC*, where priority is assigned to *lanes*. Thus, *SP-NoCs* statically reserve NoC resources at design time to certain types of flows (e. g., the lane L2 is reserved for the flow with priority 1). In *DP-NoCs* such reservation does not exist.

In *DP-NoCs*, lane priority varies according to the packet priority. This allows: (*i*) transmitting packets through any lane; (*ii*) transmitting packets through different lanes along the packet path; (*iii*) transmitting packets with the same priority through different lanes in the same physical link using time division multiplexing (TDM). In *DP-NoCs*, the priority field is also included in the packet header, being possible to assign to this field any value between zero and (2^t-1), where *t* is the flit width.

DP-NoCs keep the same external router interface of previous designs, requiring modifications in arbitration, routing table and scheduling. The arbitration method serves the packets with higher priority first, as in *SP-NoCs.* However, as the priority is

[2] Scheduling defines which flow can use a given output port.

defined in the packet, it is necessary to compare the priority field of all incoming packets, increasing the time to perform arbitration by two clock cycles. The routing table is extended with a new field, named *priority*. This field determines the priority of the output port lanes. The scheduling policy verifies the packets waiting to be transmitted to the free lanes in decreasing priority order. A round-robin algorithm is used to solve conflicts when packets with the same priority dispute the same lane. In *SP-NoCs*, this kind of conflict never arises.

3.4 Circuit Switching NoC – *CS-NoC*

The *Circuit switching* NoC adds differentiated services by enabling connection establishment. The network offers a guaranteed throughput (GT) service to flows with QoS requirements. To flows without QoS requirements, the network offers a best effort (BE) service. This approach, GT plus BE, is similar to the one implemented in the Æthereal NoC.

This design employs two lanes, L1 and L2. Lane L1 carries circuit switching data, while lane L2 is used to transmit packet switching data. GT flows have priority higher than BE flows, with end-to-end latency guarantee. When a given GT flow leaves the physical channel idle, BE flows may use this channel, without incurring in any significant penalty to GT data arriving while a BE flow is using the channel.

The physical interface between routers in CS-NoC has all signals of the previous NoC, plus an additional signal, *ack_in* (and *ack_out* respectively), to signal connection establishment (*ack_in* asserted) and connection release (*ack_in* unasserted).

A GT flow requires connection establishment before starting data transmission. A connection between a source and a target node require the reservation of lane L1 along the path between their respective routers. The path reservation avoids the establishment of other connections in the same path.

The hardware to implement circuit switching is simpler than in packet switching, since a register can be used instead of a buffer, and the control flow is simplified, requiring neither handshake nor credit control. Some NoCs, such as Æthereal, store in a table data such as the required bandwidth of the GT flows, but this table increases router area significantly. Multiple flows may use the physical channel, multiplexing the bandwidth (TDM). In the CS-NoC only one connection can be established per physical channel, not requiring this additional area. A specific protocol is used to establish and release connections. In summary, connections are established or released using BE control packets. These packets are differentiated from BE data packets by the most significant bit of the first header flit. When this bit is asserted, the BE packet has control function, and the second flit indicates the command to be executed (connection establishment or release). BE control packets do not contain payload.

4 A NoC Supporting Rate-Based Scheduling – *RB-NoC*

This Section proposes the design of a router with a built-in mechanism which overcomes circuit switching and priority scheduling limitations stated in Section 3. BE flows are transmitted using a single specific VC per channel, while QoS flows may use any VC. This resource reservation for QoS flows is needed to avoid that multiple BE flows momentarily block some channel for a QoS flow.

Telecom networks have employed rate-based scheduling policies to control congestion. Examples of such policies are virtual clock (VC) [21], weighted fair queuing (WFQ) and the method proposed in [22]. The rate-based scheduling policy proposed here comprises two steps: 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. It 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 networks. This virtual connection corresponds to a line in the *flow* table (see **Fig. 3**) 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.

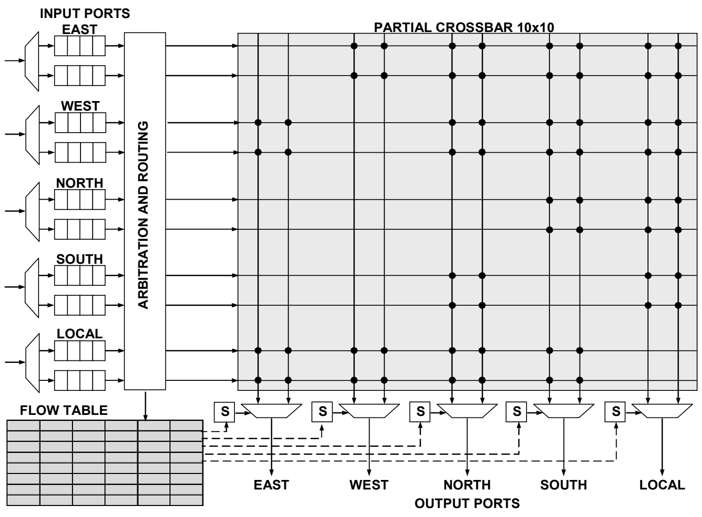


Fig. 3. Router architecture with support for rate-based scheduling.

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. 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 and routed to an output port (**Fig. 3**). Packets assigned to the same output port are served according to the proposed scheduling policy.

In the implemented scheduling policy, BE flows are transmitted only when no QoS flow requires the physical channel. The RB-NoC employs a notion of priority to differentiate QoS flows among them, but priority is defined in a different way from SP-NoC and DP-NoC definitions. In an RB-NoC, a QoS flow *priority* is the difference between the required rate and the rate currently used by it. When two or more QoS flows compete, the higher priority flow is scheduled first.

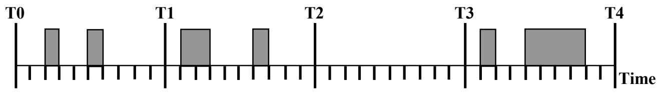
As illustrated in **Fig. 3**, the flow table is read by the scheduler (blocks named S in **Fig. 3**) to find the priority of each QoS flow assigned to a same output port. The flow priority is periodically updated according to Equation 2. A positive priority means that the flow used less than its required rate in the considered sampling period. A negative priority means that the flow violates its rate in the sampling period.

$$priority_i = required\,rate - used\ rate_i\,,\quad \text{where } i \text{ designates a given flow} \qquad \mathbf{(2)}$$

The required rate is fixed during the admission control step. The used rate (UR) is *periodically* computed according to Equation 3, where *CR* is the *current rate* used during the current period, and *UR* is the average of the previous used rate and the current used rate.

$$UR_i = \begin{cases} CR_i, & if\ UR_{i-1} = 0 \\ \dfrac{UR_{i-1}+CR_i}{2}, & if\ UR_{i-1} \neq 0 \end{cases} \qquad \mathbf{(3)}$$

Fig. 4 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 shows the behavior of one flow, 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**

Fig. 4. 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 3, 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 2.

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 physically 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 4). It can be observed that applying *CUR* to *UR* each *n* intervals (4 in this example), the error is minimized.

$$CUR = \frac{CR2}{n} = \frac{\sum_{i=0}^{n-1} CR_i}{n} \quad \textbf{(4)}$$

Consequently, in Equation 3, UR_i receives *CUR* when *i mod n* is zero, where *n* corresponds to the result of dividing the longer sample interval value by the shorter. If the used rate is considered alone in the priority computation ($priority_i = 100 - UR_i$), the scheduling policy tends to balance physical channel use, which implies disregarding that distinct QoS flows may require distinct rates, and should thus be avoided.

5 Experimental Results

The behavior of a network depends on its architecture as well as on the running application. For example, in some applications (e.g. streaming) long messages may dominate, while in others (e.g. controllers) short messages dominate traffic characteristics. According to [4], the influence of traffic in system performance is greater than that of network structural parameters. Thus, it is important to dispose of traffic generators to model the behavior of real traffic. This Section shows experiments comparing the performance of the described NoCs through functional VHDL simulation. The parameters for all NoCs are: 8x8 mesh topology; XY routing; 16-bit flits; 2 virtual channels; 8-flit buffers associated to each input lane. An 8x8, 64-router NoC is big enough to provide significant results on which to draw conclusions about future SoC interconnects, while allowing reasonable RTL simulation time. The flit with has no influence here on the QoS behavior. Thus, a close to minimum value was chosen. Buffer sizing is a complex subject, but previous experiments [19] have showed that 8 is a minimum size that does not impair NoC performance.

5.1 Experimental setup

Table 1 presents the flows used in the experiments. Flow *A* is characterized as a CBR (constant bit rate) service, i. e. a flow transmitting at a fixed rate [23]. Flow *B* is a variable bit rate (VBR) service [23]. This flow is modeled using a Pareto distribution [24]. According to [24], Pareto distributions are observable in bursty traffic between on-chip modules in typical MPEG-2 video and networking applications. Flows *A* and *B* have QoS requirements of latency and jitter. Nodes generating flows *A* and *B* transmit 2000 packets. The results do not take consider the first 100 packets, which correspond to the warm-up period. Also, the last 100 packets are discarded from results, since the traffic by end of simulation does not correspond to regular load operation. Flow *C* is a BE flow, also modeled using a Pareto distribution. This flow disturbs flows with QoS requirements (*A* and *B*), being considered as *noise* traffic. For this reason, results for the *C* flow are not discussed.

Table 1. Characterization of the flows used in the experiments.

Type	Service	QoS	Distribution	Number of Packets	Packet Size	Target
A	CBR	Yes	Uniform (20%)	2000	50	Fixed
B	VBR	Yes	Pareto (40% in the ON period)	2000	50	Fixed
C	BE	No	Pareto (20% in the ON period)	Random	20	Random

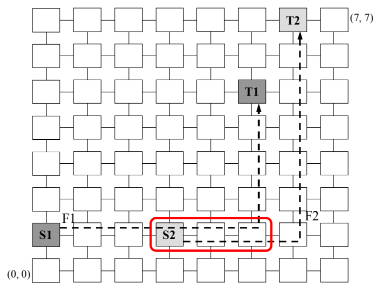
All simulations scenarios were repeated for different amounts of packets per flow (100, 200, 1000 and 2000) and different packet sizes (50 and 500 flits). The same results were observed for latency, throughput and jitter for every experiment counting 200 flits per flow or more. This means that the network reaches a steady state in this situation. Results for long packets (500 flits) are proportional to the results for short packets (50 flits). From the results included below it is easy to infer other behaviors.

Two evaluation scenarios are defined. In the first, two QoS flows (F1 and F2) originated at different nodes share part of the path to targets. In the second, three QoS flows (F1, F2 and F3) generate traffic, all sharing part of the path. All remaining network nodes transmit disturbing C flows. **Fig. 5** presents the spatial distribution of source and target nodes. The placement of source and target nodes aims to evaluate situations where the flows with QoS requirements compete for network resources.

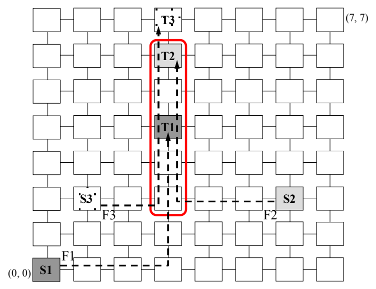
Spatial traffic distributions and the experimental scenarios were chosen to highlight the limitations of priority scheduling and circuit switching when resources are shared among flows. Models CBR (e.g. non-compacted video) and VBR (MPEG) are artificial but relevant workload models [10]. Equation 5 gives the minimal latency to transfer a packet from a source to a target, in clock cycles.

$$minimal\ latency = (R \times N) + P \qquad \textbf{(5)}$$

In this Equation: (*i*) R is the router minimal latency (arbitration and routing), equal to 5 for the BE, SP, DP and CS NoCs; for the RB-NoC this value is 13; (*ii*) N is the number of routers in the path; (*iii*) P is the packet size. **Table 2** summarizes the conducted experiments. Column Priority (P) has no meaning for BE-NoC and RB-NoC. In the CS-NoC, flows with P=1 are GT flows and flows with priority 0 are BE flows.



(a) Scenario I – two QoS flows competing for resources



(b) Scenario II – three QoS flows competing for resources

Fig. 5. Spatial traffic 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. All other nodes transmit *C* flows, disturbing the QoS flows.

Table 2. Experimental scenarios. NA stands for Not Applicable.

Experiment	Traffic Distribution	Flow F1 (QoS)		Flow F2 (QoS)		Flow F3 (QoS)		Noise flows (BE)	
		Type	P	Type	P	Type	P	Type	P
I	I	A (CBR)	1	A (CBR)	0	NA	NA	C	0
II	I	A (CBR)	1	A (CBR)	1	NA	NA	C	0
III	I	B (VBR)	1	B (VBR)	1	NA	NA	C	0
IV	II	A (CBR)	1	A (CBR)	1	A (CBR)	0	C	0
V	II	B (VBR)	1	B (VBR)	1	B (VBR)	0	C	0

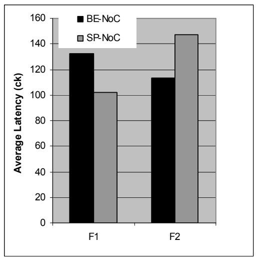
P = Priority NA = Not Applicable

The number of virtual channels (VCs) defines how many flows compete for resources in the same channel. As all NoC designs have two VCs, there are three options when more than one QoS flow coexist: all flows with low priority (using BE-NoC), some flows with high priority (I, IV and V for SP-NoC, DP-NoC, CS-NoC), or all flows with high priority (II and III for SP-NoC, DP-NoC, CS-NoC).

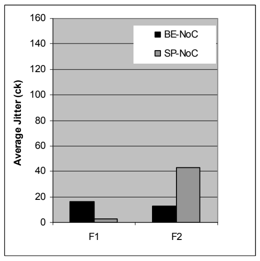
5.2 SP-NoC priority mechanism analysis

This Section compares SP-NoC to BE-NoC with regard to latency, jitter, latency spreading and throughput. **Fig. 6** gives the average latency and jitter for Experiment I. BE-NoC does not differentiate flows; i. e. average latency and jitter of packets depend on the transmission traffic conditions. Thus, BE-NoC gives no guarantees to any flow.

In SP-NoC, the highest priority flow F1 has average latency near the optimum minimum latency (5(10)+50) and jitter is close to zero. This occurs because F1 has higher priority and exclusive usage of the virtual channel L2. Therefore, whenever F1 has data to send, it has access to the physical channel. However, F2 is always blocked while F1 is delivering flits. F2 shows an average latency of about 50 clock cycles greater that the minimum latency and its jitter is about 40 clock cycles, representing 80% of the packet size. This experiment shows that a priority mechanism helps guaranteeing QoS, if flows with a same priority do not compete.



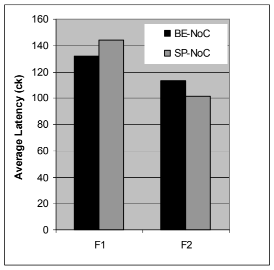
(a) Average Latency



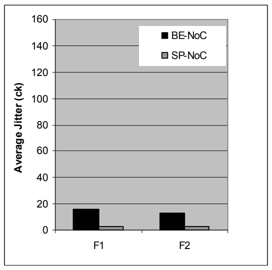
(b) Average Jitter

Fig. 6. Results for flows F1 and F2, Experiment I.

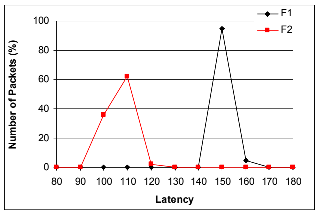
Fig. 7 shows average latency, jitter and latency spreading for Experiment II. Flows F1 and F2 have the same priority, competing for lane L2. It is noticeable that F2 has average latency near to minimum and F1 latency is around 50% larger than the minimum. This occurs because F1 and F2 are CBR flows, i. e. they insert packets in the network at fixed intervals. As F2 source node is closer to the disputed region, it is served first. For the same reason, F1 and F2 have jitter near zero and small spreading.



(a) Average latency



(b) Average Jitter



(c) Latency spreading of SP-NoC flows

Fig. 7. Results for flows F1 and F2, Experiment II, CBR traffic.

However, when F1 and F2 are VBR flows (Experiment III) results are quite different (see **Fig. 8**). Here, packets enter the network at variable intervals, using a 40% load for the ON period, representing a 20% effective load. The ON-OFF traffic model randomizes the packet injection instants. Thus, there is no flow always served first. This has two consequences: (*i*) the jitter of both flows increases, and (*ii*) due to the duration of the OFF periods, both latencies approach the minimum value.

Fig. 7 and **Fig. 8** show the priority mechanism behavior when flows with a same priority compete for network resources. In the case of CBR flows (Experiment II), one of the flows has unpredictable behavior, similar to a BE flow. In the case of VBR flows (Experiment III), the priority mechanism guarantees latencies close to minimum for the flows with higher priority. However, these present high values of jitter. Depending on the parameters that specify QoS for the flows, the usage of priority mechanism should be limited to specific situations, where competition among equal priority flows is avoidable or kept to a minimum.

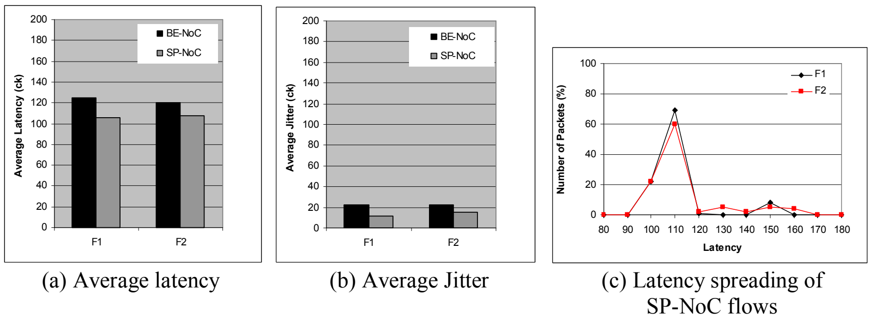


Fig. 8. Results for flows F1 and F2, Experiment III, VBR traffic.

Fig. 9 illustrates the average latency, the jitter and the throughput for Experiment IV. Here, two high priority flows compete for resources with a third low priority flow. It is possible to observe that average latency and jitter of priority flows (F1 and F2) have the same behavior of **Fig. 7**. These flows have 99% of packets with throughput between 15% and 20%, in accordance with the insertion rates. However, the low priority flow (F3) has higher average latency (about 2.5 times the minimum latency (5(8)+50) and highest jitter.

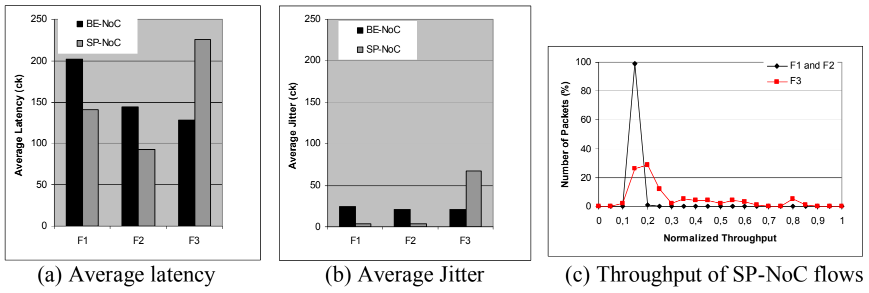
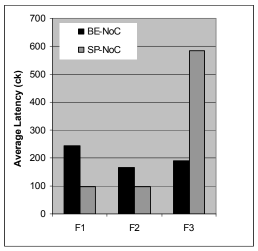


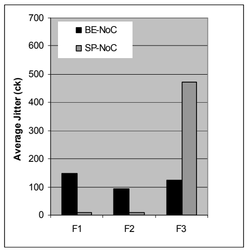
Fig. 9. Results for flows F1, F2 and F3, Experiment IV, CBR traffic.

The F3 packets throughput presents large variation, having packets with rate superior to the injection rate. This is due to the fact that packets are sent in burst after the release of the blocking condition. If F3 had some attached throughput QoS requirement, using a priority mechanism would not be adequate.

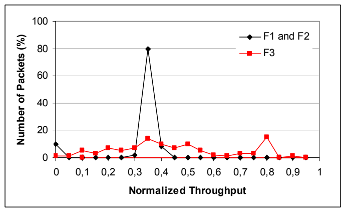
Fig. 10 shows results of Experiment V, where 3 VBR flows compete for resources. Priority flows are transmitted with near to minimum latency (90 clock cycles) and jitter close to 0. These flows have 90% of the packets with throughput between 35% and 45%, and 10% of the packets with throughput between 0% and 5%. This occurs because a VBR flow uses an ON-OFF Pareto distribution. Here, the flow with low priority (F3) is penalized, showing high latency and jitter, and erratic throughput (excessive throughput spreading).



(a) Average latency



(b) Average Jitter



(c) Throughput of SP-NoC flows

Fig. 10. Results for flows F1, F2 and F3, Experiment V, VBR traffic.

Table 3 summarizes the evaluation of the static priority mechanism.

Table 3. SP-NoC evaluation summary (all flows compete with BE packets).

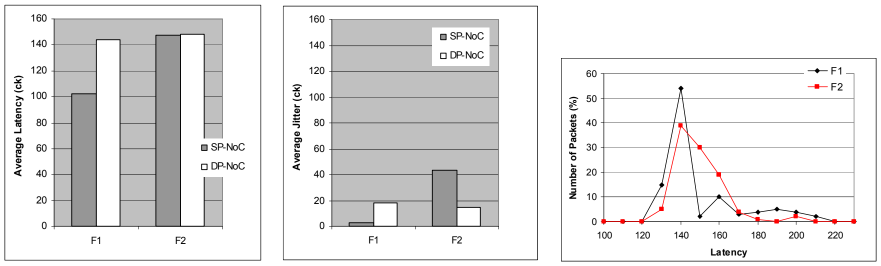
Experiment	Description	QoS guarantee			
		Latency	*Jitter*	Throughput	Reason
I	One priority CBR flow, ***without*** competition with priority flows	Yes	Yes	Yes	When the priority flows has data to transmit, it has access to the physical link.
II	Two priority CBR flows, ***with*** competition	No	Yes	Yes	The injection at fixed intervals serves first the flow which is nearer to the congestion area. Consequently, this flow has near to minimum latency, while the second flow has its latency significantly increased.
III	Two priority VBR flows, ***with*** competition	Yes	No	Yes	The injection at random intervals results in near to minimum latency, but jitter is increased.
IV	Three priority CBR flows, ***with*** competition	No	Yes	Yes	Idem to Experiment II
V	Three priority VBR flows, ***with*** competition	Yes	No	Yes	Idem to Experiment III

5.3 DP-NoC priority mechanism analysis

This Section compares *DP-NoC* and *SP-NoC*. The following performance figures are evaluated: latency, jitter and latency spreading. **Fig. 11** shows the results obtained in Experiment I, where a priority flow is transmitted without competing with any other priority flow, but competing with BE flows. The performance of the *DP-NoC* is inferior to the *SP-NoC*. Two reasons may be advanced to explain this:

1. The minimal latency for the SP-NoC is 100 clock cycles ((5*(10)+50), where 5 is the arbitration/routing delay, 10 is the number of hops and 50 is the packet length), while in *DP-NoC* is 120 clock cycles ((7*(10)+50), due to the two extra clock cycles in the arbitration/routing delay).
2. Even if the *DP-NoC* privileges higher priority flows, there is no lane reservation for priority packets. Thus, if there is no higher priority packet being transmitted, BE flows may use all lanes, blocking priority packets until a lane is freed.

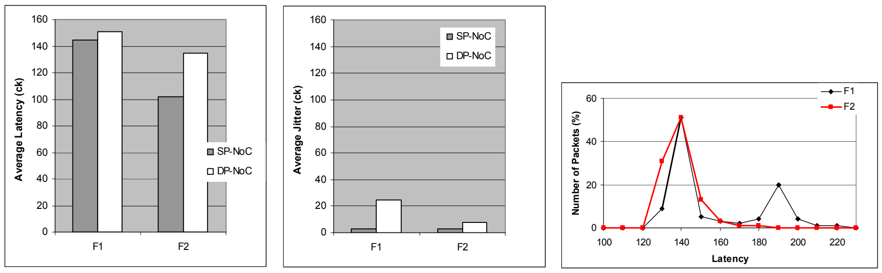
This behavior leads to unpredictable jitter values. Here, the average jitter of F2 (without priority) is smaller than F1 (with priority). The second reason advanced above leads to an important latency spreading, since priority packets are blocked.



(a) Average Latency (b) Average Jitter (c) Latency spreading ofDP-NoC flows

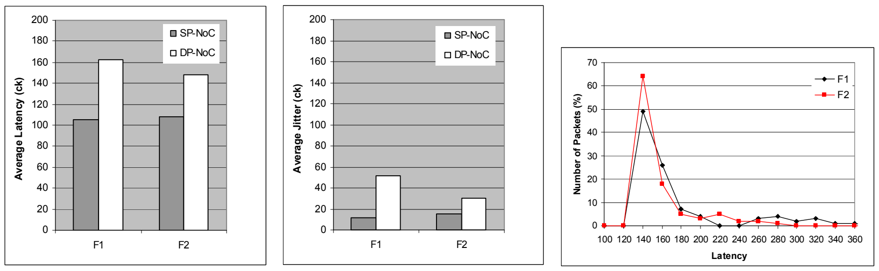
Fig. 11. Results for flows F1, F2, Experiment I.

Fig. 12 and **Fig. 13** show the results obtained when flows with the same priority compete for resources (output links). For both experiments, the performance of the *DP-NoC* is again inferior to the *SP-NoC*, for the same reason: absence of resource reservation. Increasing the number of different priorities could be effective with more lanes; however, the router area grows quadratically with the amount of lanes.



(a) Average Latency (b) Average Jitter (c) Latency spreading of DP-NoC flows

Fig. 12. Results for flows F1, F2, Experiment II, CBR Flows.



(a) Average Latency (b) Average Jitter (c) Latency spreading of DP-NoC flows

Fig. 13. Results for flows F1, F2, Experiment III, VBR Flows.

Whenever the number of flows competing for a same channel is smaller or equal to the number of available virtual channels the DP-NoC can effectively guarantee QoS requirements.

5.4 Circuit Switching Mechanism Analysis

If a QoS flow has to be transmitted without competing with other QoS flows, circuit switching mechanism is the surest way to guarantee QoS. **Fig. 14** illustrates the amount of time required for connection establishment, data transmission and connection release, using flows of Experiment II, with F1 and F2 being GT flows, competing for the same lane. The time to establish and release a connection, small in this experiment, varies with network traffic, as BE packets control these actions.

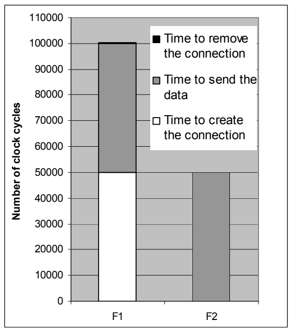


Fig. 14. Evaluation of time to connection establishment, data transmission and connection release for F1 and F2 in Experiment II using CS-NoC.

Both flows transmit 100,000 flits (equivalent to 2,000 50-flit packets of the previous experiments). As the rate for the flows is 20% of the available bandwidth, the total time to transmit all 100,000 flits is 500,000 clock cycles. As illustrated in **Fig. 14**, F2 establishes its connection first. The flow F2 spends 148 clock cycles to create the connection, plus 500,000 clock cycles to transmit data, and 73 clock cycles to remove the connection. Flow 1 waits all these clock cycles to start transmission. The, the total transmission time for both flows is approximately 1,000,000 cycles.

If packet switching is used, as in BE, SP, and DP NoCs, channels are shared among flows, resulting in a smaller time to deliver all flits (in the present case, approximately 500,000). This shows the main disadvantage of circuit switching: static reservation of resources, potentially wasting NoC bandwidth. This disadvantage can be partially minimized using time division multiplexing (TDM) to allocate the bandwidth in fixed size time slices. However, it should be noticed that regular behavior of the traffic is required when using TDM (as CBR flows) to adjust the incoming data rate to the reserved time slots. Otherwise the risk of wasting bandwidth is again present, possibly coupled to the risk of losing data.

5.5 Rate-based scheduling results

In this Section, the Rate-based NoC (*RB-NoC*) is compared to the *SP-NoC*. **Table 4** presents latency, jitter and throughput values for Experiment II (2 CBR flows with the same priority). Both scheduling policies guarantee throughput close to the inserted rate (20%). Analyzing the priority scheduling, F2 has average latency near to ideal, while F1 flow has higher latency (average latency is 77% greater than the ideal latency). Flows F1 and F2 insert packets 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 QoS when flows with the same priority compete for the same resources. In 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, bandwidth is equally divided between the flows, resulting in almost the same latency values for both flows, near to ideal values. Jitter is slightly increased when compared to priority-based scheduling, due to the higher minimal latency of the router. This result demonstrates the efficiency of the method.

Table 4. Results for flows F1 and F2, Experiment II, CBR traffic using SP-NoC and RB-NoC.

Performance Figures		SP-NoC		RB-NoC	
		F1	F2	F1	F2
Latency	Ideal (ck)	250,00	250,00	330,00	330,00
	Minimum (ck)	441,00	250,00	330,00	330,00
	Average (ck)	**443,40**	**251,86**	**333,54**	**332,42**
	Maximal (ck)	450,00	258,00	350,00	346,00
Jitter (ck)		2,14	1,78	4,07	3,01
Average throughput (%)		19,80	19,80	19,80	19,80

Table 5 displays results for Experiment III, where F1 and F2 are VBR flows. Here, packets are inserted at variable intervals, using a 40% load for the ON period. The ON-OFF traffic model randomizes packet injection instants, which inserts jitter. Thus, jitter is not showed in **Table 5**. In both scheduling methods, F1 has average latency near to the ideal one. In priority scheduling, F2 has average latency 56% higher than the ideal latency, and the rate-based scheduling only 33% higher. Despite the fact they have similar behavior, rate-based is superior to priority-based scheduling, since it is able to reduce the percentage of deviation from the ideal latency.

Table 5. Results for flows F1 and F2, Experiment III, VBR traffic using SP-NoC and RB-NoC.

Performance Figures		SP-NoC		RB-NoC	
		F1	F2	F1	F2
Latency	Ideal (ck)	250,00	250,00	330,00	330,00
	Minimum (ck)	250,00	250,00	330,00	330,00
	Average (ck)	**253,40**	**321,96**	**337,58**	**440,00**
	Maximal (ck)	266,00	390,00	477,00	545,00
Average throughput (%)		38,82	39,26	38,86	39,40

5.6 Area Results

Table 6 details the router area mapped to a 0.35µm CMOS standard cell library (TSMC). Router area is similar for the BE-NoC and the SP-NoC. DP-NoC area is superior to others, since its arbitration/routing logic needs more comparators. SC-NoC has the smallest area, since simple registers replace the input buffers of the circuit switching lane. Results point to the fact that static priority (SP-NoC) and circuit switching (CS-NoC) do not significantly increase area, compared to the BE-NoC. These mechanisms may be used to force the NoC to respect QoS requirements without increasing total area. The area for all implementations is dominated by the buffers. It is recommended to use memory generators to optimize area. Considering real IPs (with 200,000 gates), an area overhead of around 10% per IP is expected.

Table 6. Router area results targeting a 0, 35µm CMOS standard-cell library (flit size=16, 2 virtual channels, buffer depth=8), using the Leonardo synthesis tool.

	BE-NoC	SP-NoC	DP-NoC	CS-NoC
Number of equivalent gates	18,657	18,621	21,080	12,792
Estimated clock frequency (MHz)	160	168	147	175

Table 7 presents router areas obtained with the Synplify synthesis tool, targeting FPGA devices. The results follow the same proportion as the ASIC mappings, with a little area penalty for the DP-NoC, and the smallest area for the CS-NoC.

Table 7. Router area results for 2V1000 FPGA (flit size=16, 2 virtual channels, buffer depth=8).

Resource	Mapping to Xilinx XC2V1000 FPGA device								
	Used				Available	Used /Available			
	BE-NoC	SP-NoC	DP-NoC	CS-NoC		BE-NoC	SP-NoC	DP-NoC	CS-NoC
Slices	1071	1158	1383	967	5.120	20,92%	22,62%	27,01%	18,89%
LUTs	1984	2150	2529	1622	10.240	19,38%	21,00%	24,70%	15,84%
Flip Flops	513	479	646	467	11.212	4,56%	4,27%	5,76%	4,17%

The area for the RB-NoC router is not available because the HDL description is not optimized for synthesis. A small increase in area can be expected here, because only a small table and few counters were added to the NoC router.

6 Conclusions and Future Work

This work evaluated different methods to provide QoS for NoCs. Dynamic priority is inefficient to guarantee QoS, due the absence of resource allocation. Static priority and connection establishment methods may guarantee QoS. However, both present limitations, especially when flows with QoS requirements compete for network resources. As shown in Experiment I, if no flows with a same priority compete for resources, static priority mechanisms are effective. When flows with a same priority compete for resources, the static priority mechanism does not provide rigid guarantees to any of the flows. An alternative to this, increasing the number of priorities, implies increasing the amount of virtual channels, which can be prohibitive in terms of silicon area. In connection establishment methods, all QoS requirements are guaranteed after connection establishment. However, if some other flow not using connection establishment has deadlines to send data as QoS requirement then this method will be not able to guarantee this requirement.

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 proposed rate-based scheduling policy 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, with flow latencies near to ideal values. Rate-based scheduling overcomes the problem of flows with a same priority competing for resources, by balancing flows according to their required rates. With VBR traffic, where packets are randomly injected into the network, the proposed approach is also superior to priority-based scheduling. However, in this case rate-based scheduling does not currently achieves minimal latencies when QoS flows compete. One clear advantage of rate-based scheduling concerns high priority flows with differentiated QoS requirements. The experiments discussed in Section 5.5 assumed priority flows with the same throughput requirement, 20% of the available bandwidth. This was done for coherence with the other experiments. Other experiments were conducted over the RB-NoC only. In one of these, two competing CBR flows require 10 and 30% of the available bandwidth and receive 9,61 and 28,81% respectively, under the same conditions of noise traffic.

As future work it is possible to enumerate: (*i*) reducing the RB-NoC router minimal latency, responsible by increases in jitter and latency; (*ii*) evaluating the proposed method when more than three flows compete for resources; (*iii*) evaluating area overhead of the RB-NoC; (*iv*) implementing congestion control mechanisms.

7 References

[1] Rijpkema, E.; Goossens, K.; Rădulescu, A. "*Trade-offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip*". In: DATE'03, pp. 350-355.

[2] Benini, L.; De Micheli, G. "*Networks on chips: a new SoC paradigm*". Computer, v.35(1), Jan. 2002, pp. 70-78.

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

[4] Duato, J.; Yalamanchili, S.; Ni, L. "Interconnection Networks". Elsevier Science, 2002, 600 p.

[5] 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), Feb. 2004, pp 105-128.

[6] 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.

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

[8] 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.

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

[10] 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.

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

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

[14] 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.

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

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

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

[18] Bjerregaard, T.; Mahadevan, S. "*A survey of research and practices of Network-on-chip*". ACM Computing Surveys, v.38(1), 2006, pp. 1-51.

[19] 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.

[20] Mello, A.; Tedesco, L.; Calazans, N.; Moraes, F. "*Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC*". In: 18th SBCCI, 2005, pp. 178-183.

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

[22] Lee, J.W.; Kim, C.K.; Lee, C. W. "*Rate-based scheduling discipline for packet switching networks*". Electronic Letters, v.31(14), 1995,1130-1131.

[23] Kumar, A.; Manjunath, D.; Kuri, J. "*Communication Networking: An Analytical Approach*". Morgan Kaufman Publishers, 2004, 929 p.

[24] Pande, P.; Grecu, C.; Jones, M.; Ivanov, A.; Saleh, R. "*Performance Evaluation and design Trade-Offs for Network-on-Chip Interconnect Architectures*". IEEE Transactions on Computers, v.54(8), Aug. 2005, pp. 1025-1040.