BAHIA: Burst-Aware Head-of-Line Blocking Injection Avoidance

José V. Escamilla[1], José Flich[1] and Pedro J. García[2]

Abstract— **Many-core chip designs are the current manufacturing trend for high-performance computing.**

An emerging problem is on-chip network congestion, due either to several traffic classes requesting the same resources (e.g. memory controllers) or to bursty traffic interfering with other flows.

In this paper we propose BAHIA, which enables dynamic separation of bursty traffic from non-bursty one, thereby removing all the contention effects of bursts with a minimal impact on network overhead and with a marginal increase in area requirements. Results demonstrate a robust and effective split of traffic, leading to non-bursty traffic to achieve the desired low latencies even in bursty-prone conditions.

Keywords— **networks-on-chip; burst; contention;**

I. Introduction

THe high-performance computing domain is taking advantage of the inclusion of multicore solutions in the form of Chip Multiprocessor (CMP) and System-on-Chip (MPSoC) systems. As the integration scale goes further, more cores, nodes, or processing units are included in the same chip.

Both design platforms, CMPs and MPSoCs, rely on an interconnection network infrastructure that provides the communication between all the processing nodes. This must be a high-bandwidth, low-latency network to avoid slowing down processors while waiting for remote data.

As technology advances, we will have more complexity in the chip, leading to more devices. Also, specialization will drive the inclusion of dedicated devices as accelerators, encoders, DMA devices, etc. This heterogeneity and the increasing number of components will drive a change in the traffic present in those devices. We can expect bursts of traffic flowing from device to device, at intermitent and unpredictable frequencies. This kind of traffic may create temporary hotspots where the traffic is concentrated, thereby leading to the appearance of network congestion that is likely to have a negative impact on the rest of the traffic. As we show in this paper, efficiently dealing with the problems derived from congestion can significantly improve the overall chip performance.

Indeed, in this paper we present BAHIA (Burst-Aware HoL-blocking Injection Avoidance), a mechanism that dynamically detects bursty traffic in the network, then isolating the burst and thus guaranteeing that non-bursty traffic is unaffected. The BAHIA method is targetted to lightweight NoC designs, where transmission latency is of outmost importance, thus, no additional mechanisms built inside the network are added to the switches (i.e. congestion control mechanisms, traffic separation). Indeed, we follow the same approach as [1], so moving the complexity to the network interfaces (NIs).

The rest of this paper is organized as follows: Section II presents the related work. In Section III we thoroughly describe the key aspects of the BAHIA mechanism. Next, in Section IV, we detail the evaluation scenarios and the results obtained, and finally in Section V, we present some conclusions and future work.

II. Related work

In [2] and [3] the importance of bursty traffic in the congestion control framework is pointed out. In that sense, the BAHIA method described in the next section addresses the negative effects of bursty traffic.

In [4] the problems derived from bursty traffic are addressed by increasing buffer size in order to get room enough to absorb bursts. This approach is relatively expensive in terms of silicon area and power, and, as reported in the article, with non-bursty traffic this results in a suboptimal utilization of the resources. In addition, it is difficult to predict burst sizes, hence probably a burst may overflow buffers, then resulting in contention. In BAHIA, however, buffer size is not modified as bursty traffic is separated at sources.

In [5] a solution to reduce the latency in worst-case bursty traffic is proposed. However, this mechanism is based on temporarily ejecting packets and later reinjecting them with a priority-based approach. This achieves good results as it helps in the worst case, but at the cost of increasing latency of newer packets. Moreover, if we consider a scenario with several virtual networks, this mechanism requires three queues per virtual network at each node, thus becoming an expensive solution in an NoC context.

III. BAHIA Description

BAHIA (Burst Aware HoL-blocking Injection Avoidance) provides a method to, at runtime, isolate detected bursty traffic in a network, in order to prevent bursts from causing HoL-blocking. Detection of bursty traffic is performed at any end-node receiving a burst. All the end-nodes are then notified of this detection, so that thereafter the bursty traffic can be identified in order to be separated from non-bursty traffic, thereby avoiding the HoL-blocking that the

[1]Grupo de Arquitecturas Paralelas, DISCA, Universitat Politècnica de València, Valencia, Spain joseslo@gap.upv.es

[2]Escuela Superior de Ingeniería Informática, Universidad de Castilla-La Mancha, Albacete, Spain PedroJavier.Garcia@uclm.es

former could produce to the latter. BAHIA makes use of virtual networks to separate traffic, hence BAHIA requires at least two virtual networks: the "default" virtual network, for non-bursty traffic, and an extra virtual network for bursty traffic.

A. Burst detection

As mentioned above, the detection of bursty traffic is performed at the end-node receiving the burst. For that purpose, each end-node periodically calculates its rate of received traffic. The traffic rate is calculated every "polling interval" (PI) cycles, which is a predefined parameter of BAHIA. If that rate exceeds a given high-threshold (HT) value, this end-node will notify the other end-nodes that it is receiving bursty traffic. Similarly, any end-node notifying bursty traffic must be able to detect the end of the burst. For that purpose, the traffic reception rate is compared with a given low-threshold (LT) value. Accordingly, in this case, all the end-nodes will be notified.

B. Burst notification

In order to notify of a traffic burst to all the end-nodes, BAHIA makes use of a simple dedicated signaling network (Burst Notification Network, BNN). Basically, the whole BNN is a set of one-bit-wide overlapped control networks, each one managed by a specific end-node. Each control network bit connects its manager end-node to the rest of end-nodes, the former being the only one able to activate/deactivate the signal (i.e. to set the bit to one/zero) of this control network, while the latter being just signal receivers. Thus, every end-node owns an exclusive one-bit-wide control network to notify bursts events to the rest of end-nodes. Figure 1 shows the one-bit-wide control network managed by end-node 0, but note that every end-node owns a similar independent one-bit-wide control network to send notifications to the rest of nodes, so that in this 4x4 mesh network, there would be other 15 one-bit-wide control networks besides the one shown. The overlapping of these independent control networks allows every end-node to notify bursts without risk of collisions with notifications from other end-nodes.

Any end-node notifies the rest of end-nodes of a burst by setting to high value the signal of its BNN line. Due to the simplicity of that signal, it reaches all the end-nodes in a few cycles. The time spent in propagating and processing this signal is modeled by the "notification delay" (ND) parameter in the simulations. It is worth mentioning that the processing of this signal is negligible (and so the required hardware). Regarding the area overhead introduced by the BNN, in [6] a far more complex additional network is proposed, however, authors conclude that the area overhead for this network is negligible.

C. Traffic separation

Every end-node implements a "burstiness bit-vector", each bit corresponding to a specific end-node in the network, as can be seen in Figure 2.

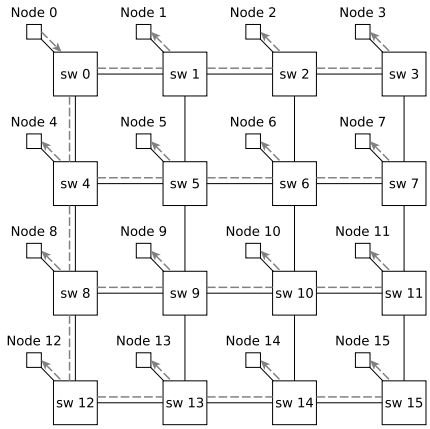

Fig. 1. Node 0 communicates burst events through this 1-bit network

When an end-node detects a high signal value in the BNN line associated to an end-node, the former will set to one in its burstiness bit-vector the bit corresponding to the latter. All the messages are mapped to the default virtual network once generated, but the destination of every message is checked before injection to obtain the value of the bit in the burstiness bit-vector corresponding to the destination end-node of the message. If that bit is set to one, the message will not be directly injected but transferred to the extra virtual network. Obviously, messages addressed to an end-node whose associated bit in the burstiness bit-vector is set to zero, will remain mapped to the default virtual network.

Figure 2 shows the basic structure of the sender part of an end-node that has messages addressed to end-nodes 1, 5 and 6. The message at the header in that queue is addressed to end-node 5. As can be seen, in the burstiness bit-vector, the bit corresponding to end-node 5 is set to one (i.e. end-node 5 previously notified that it was receiving a burst), so this message must be mapped to the extra virtual network. Indeed, before injection, the arbiter at the sender node checks the burstiness bit-vector and transfers the message addressed to end-node 5 to the queue associated to the extra virtual network, so that this message will be later injected from that queue.

It is worth pointing out that, although messages are injected either from the default queue or from the extra one, all of them are initially mapped to the default VN. This is because, if an end-node directly maps to the extra VN the messages addressed to an end-node that has recently notified a burst, there may be messages still stored in the default queue that are addressed to the same destination, and this could introduce out-of-order message injection (and so delivery) as queue selection policy is based on a simple round-robin algorithm. Hence, to guarantee in-order message injection and delivery, all the messages are first mapped to the default VN and the arbiter is provided with some additional intelligence to check the burstiness bit-vector, in order to eva-

luate whether a message should be directly injected from the default queue or, on the contrary, it should be transferred (by changing pointers) to the extra virtual network.

Once an end-node notifies that it is no longer receiving bursty traffic (by setting to low value the signal of its BNN line), the other end-nodes will reset the corresponding bit in their burstiness bit-vector. Thereafter, new messages addressed to this end-node will be injected from the queue associated to the default virtual network. However, note that this may also introduce out-of-order message injection and delivery, as messages addressed to the end-node may remain in the extra queue. The example of Figure 2 also shows a situation where there are messages in the extra queue addressed to an end-node (specifically, end-node 0) whose associated bit in the burstiness bit-vector has changed from one to zero.

In these cases, in-order packet delivery can be preserved if messages addressed to a specific end-node are injected from the default queue only if there are no messages addressed to the same destination in the extra queue; otherwise, the packet must be transferred from the default queue to the extra one. However, this implies that other information than that of the burstiness bit-vector is necessary, besides some additional tasks for the arbiter. Specifically, every end-node in BAHIA implements a presence vector. This vector contains an element per end-node in the network and every element is a counter indicating how many messages adressed to this end-node are stored in the extra queue. Every counter is incremented each time a message addressed to the corresponding end-node is moved from the default VN to the extra one, and decremented when messages are injected to that end-node from the extra VN. When a message reaches the head of the default queue and the bit associated with its destination in the burstiness bit-vector is set to zero, the counter associated with that destination in the presence vector is also inspected: if the value of that counter is zero, the message is injected from the default queue; otherwise, the message is moved to the extra queue.

The whole mechanism necessary to keep in-order message injection is a post-processing mechanism, in the sense that messages are mapped to their final VN once they reach the head of the default queue, and not before. As mentioned above, the arbiter should be in charge of performing this post-processing mechanism, that can be summarized in the next pseudocode:

```
rr=true
while(true) {
  if(rr && !isDefaultVNempty()) {
    if(isNodeReceivingBurst(msg.
        destination) || numFlitsInExtraVN(
        msg.destination) > 0)
      moveMessageToExtraVN(msg);
    else
      injectFromDefaultVN();
  } else if(!isExtraVNempty())
    injectFromExtraVN();
  rr=!rr
}
```

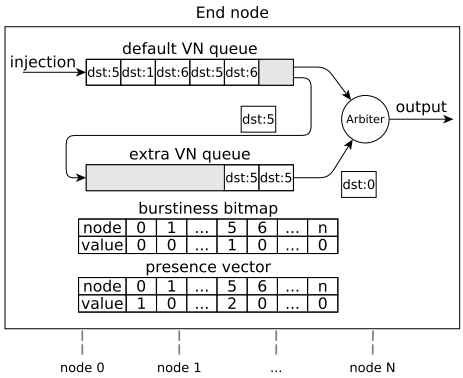

Fig. 2. Flow followed by messages in a node

IV. Evaluation

In this section we present a preliminar evaluation of BAHIA based on the results of simulation experiments performed in bursty traffic scenarios. First, we describe the simulation environment. Second, we offer a robustness analysis of the diferent parameters that define the BAHIA behavior. To carry this out we stablish a baseline configuration, then running simulations with different values of each parameter, in order to find out the parameters of the configuration that optimizes BAHIA performance. Third, we compare the results obtained with and without BAHIA in the same scenario. Fourth, we perform an analysis of the influence of the number of virtual networks over BAHIA and the performance improvement regarding similar no-BAHIA scenarios. Finally we perform analysis varying traffic patterns.

In order to better appreciate the BAHIA effect on the traffic, results are shown in two graphs: one containing default virtual network(s) data and one containing the extra virtual network data. Notice that we do not target overall throughput increase or overall latency reduction. Our aim is to separate bursty traffic from non bursty one, thereby keeping the latter unaffected by the HoL-blocking the former may produce.

A. Simulation environment

An NoC simulator has been used for the experiments. Results are shown every 500 simulated cycles. The network topology modeled in all the experiments is a 2D mesh built from 64 switches arranged in a 8x8 mesh distribution, each switch being connected to a single end-node. Regarding the traffic pattern, each end-node generates baseline/background traffic following an uniform pattern and at a rate of 0.2 flit/cycle. In addition, when simulation reaches cycle 10000, traffic bursts are sent to 4 differents nodes following a 4-to-1 strategy (i.e. each hotspot end-node will receive traffic from 4 nodes) at a rate of 1 flit/-cycle until cycle 20000 is reached. Meanwhile, the rest of nodes keep sending at a rate of 0.2 flits/cycle.

TABLE I

BAHIA ROBUSTNESS ANALYSIS CONFIGURATION

	HT (flits/cycle)	LT (flits/cycle)	PI /cycles)	ND (cycles)
baseline	0.7	0.2	500	4
HT analysis	0.5, 0.6, 0.7, 0.8, 0.9	0.2	500	4
LT analysis	0.7	0.1, 0.2, 0.3, 0.4, 0.5	500	4
PI analysis	0.7	0.2	100, 200, 400, 800, 1600	4
ND analysis	0.7	0.2	500	1, 2, 8, 16

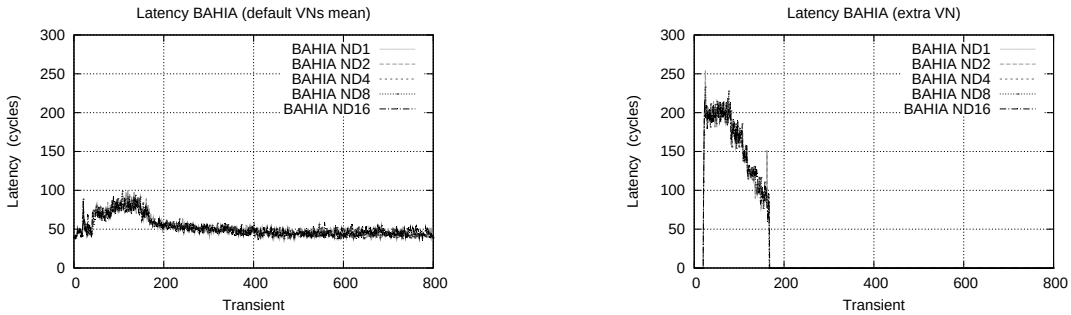

Fig. 3. Notification delay analysis.

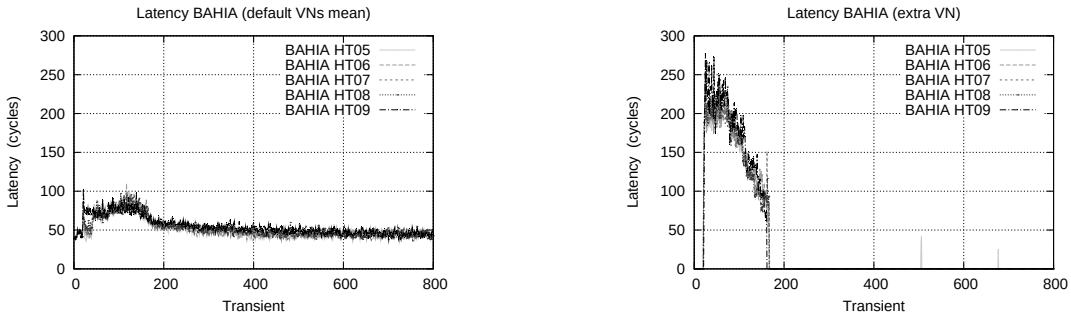

Fig. 4. High-threshold analysis.

B. Parameters tuning

BAHIA behaviour is defined by four parameters: BNN notification delay (ND), high-threshold (HT), low-threshold (LT) and polling interval (PI). In order to explore the robustness of the mechanism, we have carried out an analysis of every parameter by independently simulating different variations of the parameters.

The baseline configuration, and the different values of the parameters used are shown in Table I.

First, we analyze the effect of varying the delay of the BNN network. In Figure 3 we can see the results. This figure shows how notification delay has negligible effects on the BAHIA behaviour. This gives some reliability against unexpected jitter delays and grants flexibility for hardware implementation since BAHIA has no strict delay requirements. As can be seen the traffic through the default VN is slightly affected by the injection of bursty traffic regardless the value of notification delay. Thus, having an aggressive BNN network with one cycle-delay or having one with 16 cycles delay does not significantly affect the results.

Next we analyze the effect of varying the value of the detection (high threshold at the receiver side of the end-nodes). Figure 4 shows results for different values of the detection threshold (HT), ranging from 50% to 90% of traffic reception rate. Note that, the lower the threshold, the more aggressive the mechanism (i.e. more sensitive to traffic bursts), but it may incur in false positives (i.e. non-bursty traffic detected as bursty). By contrast, the higher the threshold, the more selective the mechanism and it may not detect real bursty traffic. However, as we can see in Figure 5, for all the tested values of HT, BAHIA behaves similarly except for the 90% threshold value. In this case, the mechanism's response is slow. Nevertheless, in that case latency of non-bursty traffic is still kept to its bounds and the mechanism successfully detects and separates bursty traffic. Also, we can see that a low detection threshold (50%) introduces some traffic in the extra VN (false positives). However, this is not harmful as this traffic is not penalized and overall latency keeps bounded.

Similarly, Figure 5 shows the results obtained when varying the value of the low threshold (used to detect the end of bursts). As can be seen this parameter does not affect BAHIA performance because, when a burst ends, all virtual networks are free from HoL-blocking effects, so traffic flows smoothly through all virtual networks, thus no matters which virtual network a flow is mapped to.

Finally, the value of the polling interval has been

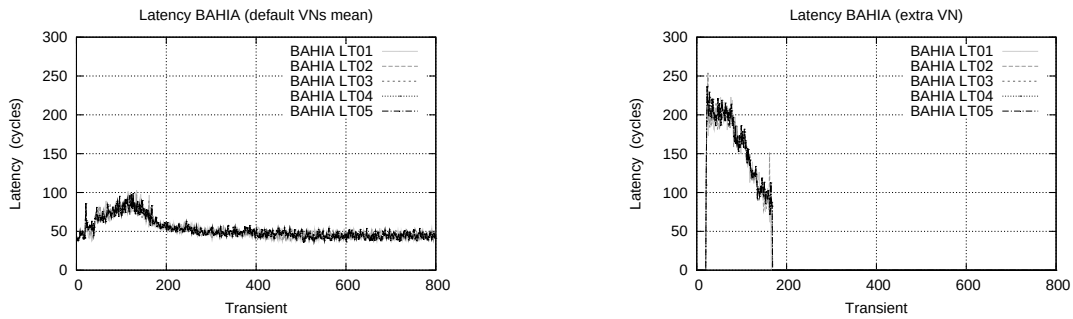

Fig. 5. Low-threshold analysis.

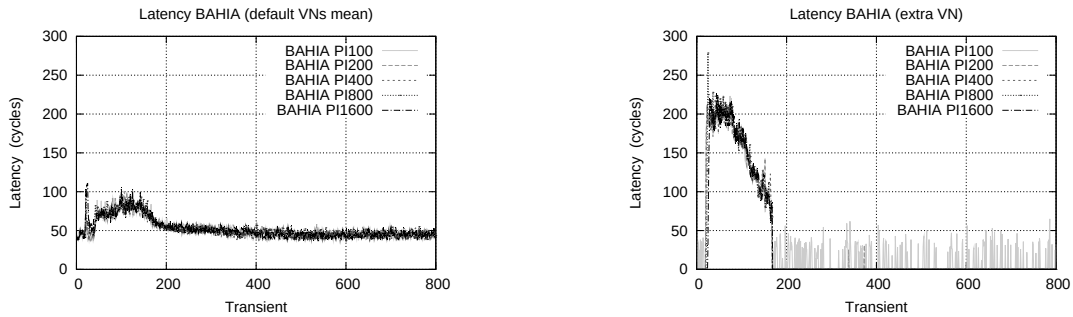

Fig. 6. Polling interval analysis.

tested. As can be seen in Figure 6, all the values but 100 cycles seem to be a correct choice, correctly reacting to the burst so that no false positives are detected. Nevertheless, the 100-cycle PI value, despite of their false positives, achieves a lower latency in the extra virtual network since it reacts more quickly to the burst. Thus, taking into account that false positives have no harmful effects, it may be a correct choice. Note that the polling interval tuning presents a close relationship with the high-threshold analysis: having a small polling interval is similar to having a lower HT value. The contrary also applies: a large polling interval could lead to the mechanism filtering short transient bursts. We conclude, then, that a small polling interval is convenient and not harmful.

C. BAHIA vs no-BAHIA analysis

In this section we carry out a general analysis of BAHIA in comparison with similar scenarios without BAHIA in order to quantify the performance improvement of BAHIA. In the previous section we have delimited optimal values for the parameters that defines BAHIA behavior for the scenario used in the simulations, so in this analysis we have set BAHIA parameters according to these optimal values.

C.1 Baseline analysis

For this analysis, we have performed simulations for the baseline configuration of BAHIA. Figure 7 shows the latency achieved with and without BAHIA. Note that, for the extra VN, there is no data corresponding to no-BAHIA as in this case there is no extra VN, all the virtual networks being analogous to the default virtual networks in the BAHIA case. In Figure 7 we can see that for the no-BAHIA case, HoL-blocking appears in the VN queues resulting in high network latency. On the other hand, for the BAHIA case, we got a significant improvement since we have reduced network latency approximately a 120% in the default VN.

C.2 Virtual networks analysis

Now, we turn our attention to the scenario when different sets of virtual networks are available. We analyze 2, 4, and 8 virtual networks. For the BAHIA case we keep a single extra virtual network and the remaining virtual networks are used as normal queues.

Figure 8 shows the results. As can be seen, the higher the number of virtual networks the higher the latency increase in the no-BAHIA case. In contrast, when using BAHIA, non-bursty traffic latency keeps bounded and minimized, reaching high latency reductions up to a factor of 15x.

C.3 Traffic pattern analysis

This analysis consists in changing the hotspot patterns. Specifically, we generate fewer hotspots but more intense ones, as traffic bursts are generated by creating 2 hotspots with an 8-to-1 strategy.

In Figure 9, we can see results for the 2-hotspot simulations. As we have 16 end-nodes sending at a rate of 1 flit/cycle of data rate to only two end-nodes, contention is increased. As can be seen, BAHIA achieves good results isolating non-bursty traffic in the default virtual network.

V. Conclusions

It is well-known that HoL-blocking considerably degrades the overall network performance. In fact, this harmful effect is pronounced as more and more

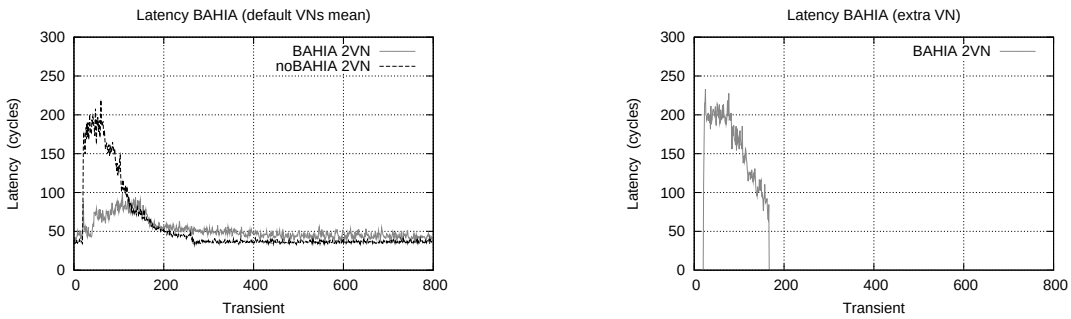

Fig. 7. Latency for the BAHIA vs no-BAHIA analysis.

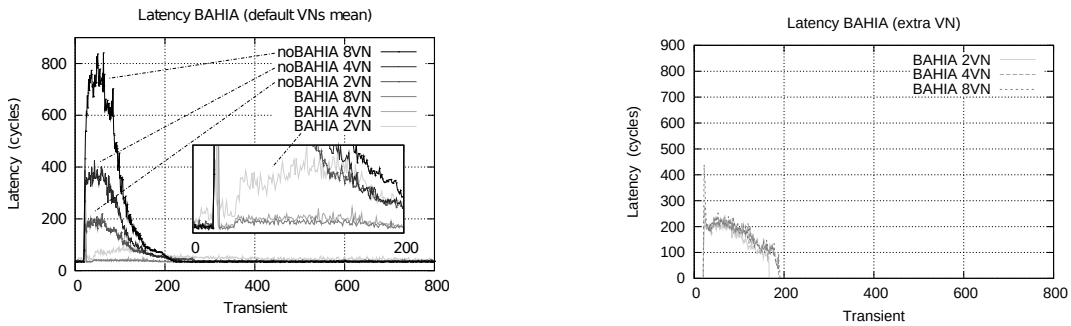

Fig. 8. Latency for the VNs analysis.

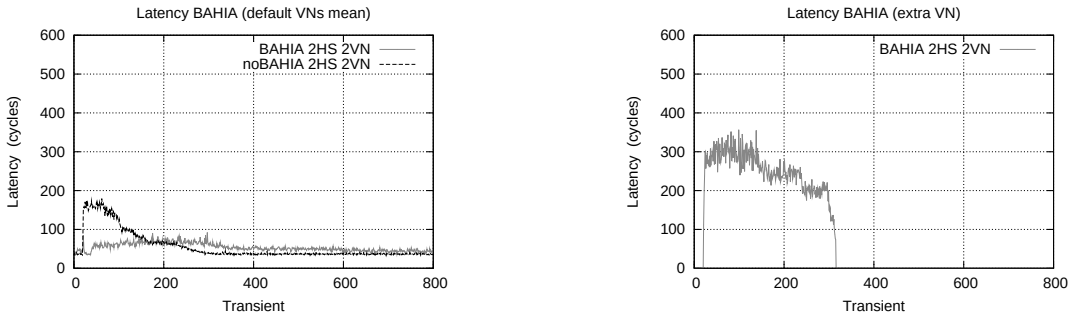

Fig. 9. Latency for the 2-hotspots analysis.

virtual networks are available in the system. In this paper we have presented a solution (BAHIA) to solve this problem by isolating bursty traffic from non-bursty one. With this mechanism we do not address congestion directly but we avoid HoL-blocking. Thanks to this, we achieve good results in keeping latency and throughput of regular traffic, obtaining up to fifteen times better latency when compared with the same scenario without BAHIA. As future work we plan to deal with internal congestion within the network and solve the problem in the same way, by eliminating at runtime the HoL-blocking introduced by congested traffic.

Acknowledgment

This work has been supported by the project NaNoC (grant agreement no. 248972) which is funded by the European Commission within the Research Programme FP7 and the Spanish MICINN, Plan E funds, under Grant TIN2009-14475-C04-01.

References

[1] Marcello Coppola, Miltos D. Grammatikakis, Riccardo Locatelli, Giuseppe Maruccia, and Lorenzo Pieralisi, *Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC*, CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2008.

[2] J.D. Owens, W.J. Dally, R. Ho, D.N. Jayasimha, S.W. Keckler, and Li-Shiuan Peh, "Research challenges for on-chip interconnection networks," *Micro, IEEE*, vol. 27, no. 5, pp. 96 –108, sept.-oct. 2007.

[3] R. Marculescu, U.Y. Ogras, Li-Shiuan Peh, N.E. Jerger, and Y. Hoskote, "Outstanding research problems in noc design: System, microarchitecture, and circuit perspectives," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 1, pp. 3 –21, jan. 2009.

[4] Tobias Bjerregaard and Shankar Mahadevan, "A survey of research and practices of network-on-chip," *ACM Comput. Surv.*, vol. 38, June 2006.

[5] M. Millberg and A. Jantsch, "Priority based forced requeue to reduce worst-case latencies for bursty traffic," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, april 2009, pp. 1070 –1075.

[6] Alberto Ghiribaldi, Daniele Ludovici, Michele Favalli, and Davide Bertozzi, "System-level infrastructure for boot-time testing and configuration of networks-on-chip with programmable routing logic," in *VLSI-SoC*. 2011, pp. 308–313, IEEE.