# Increasing the Efficiency of Latency-Driven DVFS with a Smart NoC Congestion Management Strategy

# Increasing the Efficiency of Latency-Driven DVFS with a Smart NoC Congestion Management Strategy

José V. Escamilla and José Flich
Grupo de Arquitecturas Paralelas, DISCA
Universitat Politècnica de València
joseslo@gap.upv.es, jflich@disca.upv.es

Mario R. Casu
Department of Electronics and Telecommunications
Politecnico di Torino
mario.casu@polito.it

***Abstract*—Dynamic Voltage and Frequency Scaling (DVFS) can be a very effective power management strategy not only for on-chip processing elements but also for the network-on-chip (NoC). In this paper we propose a new approach to DVFS in NoC, which combines a congestion management strategy with a feedback-loop controller. The controller sets frequency and voltage to the lowest values that keep the NoC latency below a predetermined threshold. To cope with burstiness and hotspot patterns, which may lead the controller to overdrive the NoC with too high frequencies and voltages, leading to excessive power consumption, the congestion management strategy promptly identifies the flows that caused the abnormal traffic situation and eliminates them from the latency calculation, leading to a significantly higher power saving. Compared to a baseline DVFS strategy without congestion management, our results show that our proposal saves up to 53% more power when bursty or hotspot-based traffic patterns are detected. In addition, since we also apply power-gating to make an efficient use of the network buffers, we achieve an improvement of up to 38% in power savings when no bursts or hotspots are present.**

## I. Introduction

Integrating a large number of processing elements into a single chip (CMPs, MPSoCs) is becoming the standard design choice in industry. This strategy offers good performance/power tradeoff while saves costs. These systems must be delivered with built-in networks, known as network-on-chip (NoCs) [1]. Usually, the NoC is designed with strict requirements in terms of throughput and latency so it becomes one of the most important chip components to guarantee the expected chip performance. In addition, the NoC may represent up to 20% of the overall chip power consumption [2].

The current trend is to increase the number of processing units as long as technology shrinks. Examples of this trend are the 72-cores Tile-Gx [3] or the 256-cores Kalray MPPA-256 (Bostan) [4]. With the number of processing elements increasing, it is mandatory to use strategies that reduce overall power consumption without affecting significantly system performance. One of the most successful mechanism to perform this is DVFS [5], which drives voltage and frequency dynamically at runtime to fit the workload requirements.

DVFS-based mechanisms essentially collect metrics from the system to find out how it is performing. According to this, the system reacts by increasing or decreasing frequency and voltage to meet the system requirements, thus saving power when requirements are low. The application of this mechanism to the NoC is not trivial. One main issue is to find the frequency that fits the whole NoC requirements. For small systems or systems with a very regular workload, it may become trivial. However, in larger or non-balanced workloads, that target may be difficult to achieve since some parts of the network may be overloaded while the rest of the network is completely underutilized.

On the other hand, an emerging trend is, instead of using several identical cores, to manufacture heterogeneous chips [6]. This paradigm is based on the fact that specialized processing units are more efficient at performing specific jobs. However, due to this heterogeneity the network load becomes very unbalanced and unpredictable, characterized by hotspots-based traffic patterns [7]. In addition, some application traffic patterns may naturally generate abrupt traffic bursts [8] and generate congested network regions.

Because of these reasons, it becomes apparent that guaranteeing acceptable performance levels while reducing power consumption is a real challenge. To illustrate this issue, in Figs. 1-3 we can see how a system that uses the NoC latency as metric to drive the DVFS mechanism (DMSD) [9] behaves under a critical traffic pattern. This pattern mixes a *background* traffic generated by regular nodes in a 8x8 2D mesh (e.g. general purpose processing units) with a *hotspot* traffic injected by four nodes towards a single node. This hotspot is representative of traffic generated by device hardware accelerators [10] or irregular traffic generated by some applications [8], both characterized by short and heavy-weight data bursts from/to neighbor nodes. Specifically, the background traffic is generated and received by all nodes not belonging to the hotspot following an uniform distribution pattern. Accordingly, the hotspot traffic is generated by injecting traffic at a high data rate to a given node from all of its neighbors.

The results reported in Figs. 1-3 have been obtained with a cycle-accurate NoC simulator which models 4-stages pipeline routers: IB (input buffer), RT (routing), VA/SA (virtual channel and switch allocation), X (link crossing). In Tab. I the rest of configuration parameters are described. The values in Tab. I are used to obtain our baseline results[1]. To obtain our power results we used a modified version of Orion v3.0 [11].

[1]We also evaluated the robustness of our solution when some of these parameters are varied, as we will see in Sec. III-E.

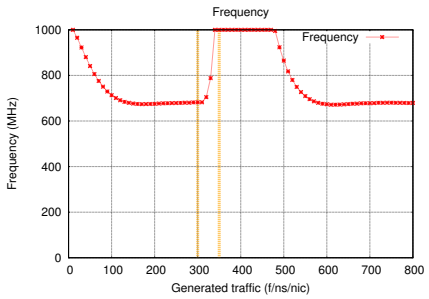


Fig. 1: Frequency for DMSD under hotspot traffic.

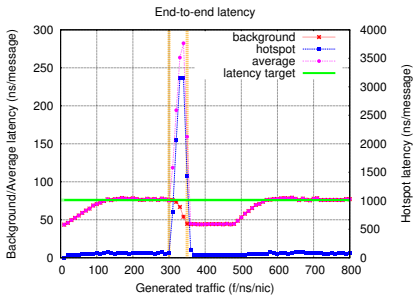


Fig. 2: End-to-end latency per traffic type for DMSD under hotspot traffic.

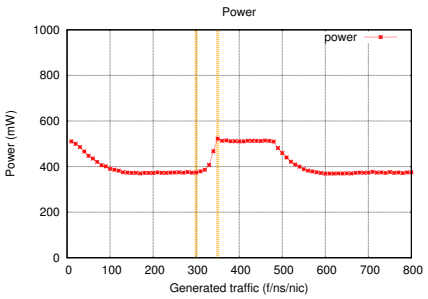


Fig. 3: Power consumption for DMSD under hotspot traffic.

Fig. 1 shows the frequency increase as the hotspot is activated at 300 $\mu s$. Fig. 2 shows how this increase differently affects the latency of two different traffic classes: congested traffic (hotspot traffic) and regular traffic (background traffic). The regular traffic is unnecessarily accelerated and its latency becomes less than a predetermined target (highlighted with a horizontal green dashed line), whereas the latency of the congested traffic increases significantly in spite of the high NoC clock frequency. As shown in Fig. 3, the consequence is a net power waste for an unwanted decrease of latency of the real productive traffic (the background one in our example).

In this paper we tackle such problem. We combine a latency-driven DVFS strategy (DMSD, as proposed in [9]), with a congestion management mechanism (ICARO [12]). Our goal is to use the congestion management strategy to discriminate and separate both traffic types, allowing the network to apply frequency and voltage policies based on the real productive traffic, hence allowing the DVFS strategy to guarantee a given end-to-end latency while optimizing power consumption.

The paper is organized as follows. First, we briefly describe DMSD and the methodology we used to obtain our results. Then, we describe ICARO. After presenting the combined strategy and its internal arrangement, we show the results. Then we revisit the related work. Finally we plot the conclusions and the future work plans.

## II. Analysis of the DMSD DVFS Policy

The purpose of the *Delay-based Max Slow Down (DMSD)* DVFS policy is to decrease the NoC frequency and voltage as much as possible without compromising the system performance [9]. To achieve this, DMSD uses the average end-to-end latency as a performance metric, and a Proportional-Integral (PI) controller that adapts frequency and voltage so as to keep that metric close to a *latency target*. The higher the latency target, the larger is the power saving. In our experiments, we set it equal to the latency that is obtained with an injection rate 5% less than the saturation point under uniform traffic, which is obtained by running simulations in which the injection rate is increased until saturation. However, in a real system the latency target can be obtained by means of profiling.

In Fig. 4 an overview of an NoC provided with DMSD is depicted. Each node stores in a register the average end-to-end latency, updated each time a flit is received. Periodically,

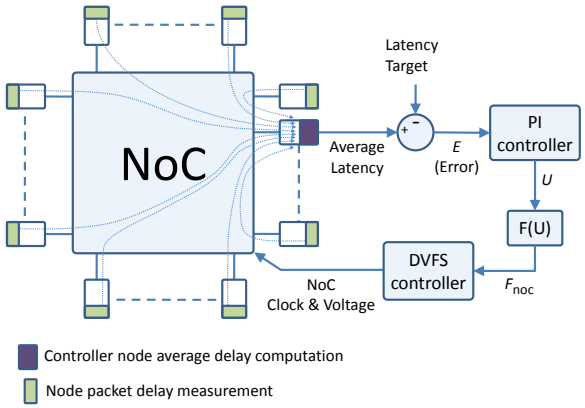


Fig. 4: All nodes in the network send latency measures to the PI controller to set the new frequency.

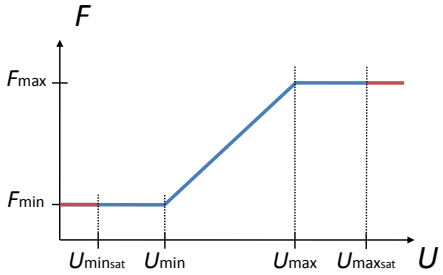


Fig. 5: Conversion from U to frequency.

all nodes send the average latency to a given node, which computes the overall end-to-end latency for the whole system. In addition, this node contains the PI controller and the voltage and frequency controllers. Upon receiving all latencies, the overall latency at time $n$, $L_n$, is computed and the noise filter described by (1) is applied to obtain $L'_n$. Then, the *error* $E_n$ is computed by subtracting the *latency target* $L_t$ from $L'_n$ as shown in (2). The error is then passed to the PI controller, which generates the signal $U_n$ according to (3).

$$L'_n = \alpha\, L'_{n-1} + (1-\alpha)L_n \quad (1)$$

$$E_n = L'_n - L_t \quad (2)$$

$$U_n = U_{n-1} + K_I E_n + K_P(E_n - E_{n-1}) \quad (3)$$

In (3), $K_I$ and $K_P$ are the integral and proportional gains determined empirically and used to adjust the PI controller

TABLE I: Robustness analysis scenarios configuration.

| **Network configuration** | |
|---|---|
| Topology | 8x8 2D regular mesh |
| Routing policy | XY |
| Switching technique | Wormhole (flit-level) |
| Flow control | credits |
| Flit size | 128 bits |
| Message size | 10 flits |
| Switch queue size | 4 flits |
| Virtual Channels | 4 per Virtual Network |
| **DMSD configuration** | |
| Frequency range | 333 - 1000 MHz |
| Voltage range | [0.56, 0.9] V |
| $K_i$, $K_p$ | 0.025, 0.0125 |
| $U$ saturation range | [-15, 15] |
| $\alpha$ | 0.7 |

behavior while guaranteeing stability.

Finally, $U$ is used to determine the frequency. For this, $U$ is bounded within $U_{min_{sat}}$ and $U_{max_{sat}}$ and the range from $U_{min}$ to $U_{max}$ is linearly translated to frequency, as shown in Fig. 5. A voltage-to-frequency mapping is then used to apply the correct voltage for a given clock frequency.

DMSD performs well under stationary traffic patterns [9]. As shown in Figs. 1-3, however, the high intensity of a few data flows (hotspots) which are not representative of the whole system load, disrupts the DVFS strategy, leading to a waste of power by increasing the frequency and voltage unnecessarily.

Notice that this effect could be avoided by implementing Voltage and Frequency Islands (VFIs) [13][14]. However, this would imply extra silicon area and power to implement the VFIs separate DVFS controllers, and it would require to either know at design-time where hotspots will be located, or the ability to confine the hotspot traffic in a separate voltage island at run-time. In contrast, our approach consists in implementing a congestion control mechanism (ICARO) to detect hotspots and filter them out, regardless their location and intensities.

Orion does not directly support the industrial 28-nm CMOS technology that we used for the implementation of routers with support for congestion management and buffer power-gating. By using the post-synthesis results of our RTL version of the router, we modified Orion in such a way that its results are compatible with our technology. Moreover, we added the support for including the effect of buffer power-gating in the computation of power consumption.

## III. Implementing Congestion Management

Hotspot flows mask the overall system performance, increasing significantly the overall latency while the network resources not used by the hotspot flows may be underutilized. Our approach consists in identifying those hotspot flows, and separating them from the rest of the network traffic (background traffic). For this purpose, we pick ICARO, a congestion-control mechanism that detects, identifies, and isolates congestion within the network.

### A. ICARO

ICARO removes the Head-of-Line (HoL) blocking by first identifying congestion, and then by isolating the congested flows involved in it into dedicated Virtual Networks (VNs). As

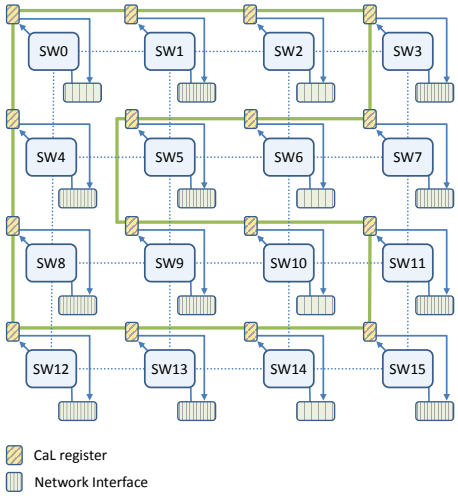


Fig. 6: Congestion Notification Network for a 4x4 mesh.

the congested traffic is delivered through separate resources, it does not share buffer resources with the non-congested traffic, so HoL-blocking is removed. ICARO consists of three stages: congestion detection, notification and isolation.

*1) Congestion Detection:* The congestion is detected at the routers level, by keeping track of which input ports are requesting any output port. If more than one input port is requesting a given output port for too much time, that output port is marked as oversubscribed, so congestion at that output port is declared. However, two or more input ports could be requesting a given output port at a low data rate, not leading to congestion. Because of this, only input ports exceeding a given utilization threshold are considered.

*2) Congestion Notification:* Once congestion is detected, it must be notified to the NIs to isolate the traffic that causes the congestion before being injected into the network. To perform this, ICARO uses a dedicated congestion notification network (named *CaL* network[2]), which consists of a ring of *N* registers connected by links of $log_2(N) + p + 1$ width, where *N* is the number of nodes and *p* the routers radix. An example of a CaL network is shown in Fig. 6. Other mechanisms for fast notification delivery include *express channels* [15] and circuit-switched networks. However, express channels may imply high area overhead while circuit-switched networks suffer from high latency penalties when establishing circuits.

*3) Congestion Isolation:* In absence of congestion, the NI allocates all messages in *regular-VNs*, as determined by the NI allocator. Once notifications are received, the NI uses the congestion information and calculates a message route to know whether that message will cross any of the *congested points (CPs)*. If so, the message is reallocated into a special VN (*extra-VN*) to be injected and delivered through it along all the path to destination. Otherwise, the message is injected through the current regular-VN. By doing this, flows contributing to the congestion are isolated into the *extra-VN*, keeping the non-

[2]This network is called CNN in [12] but we modify it to deliver also other messages, hence the name CaL (Congestion and Latency) network.

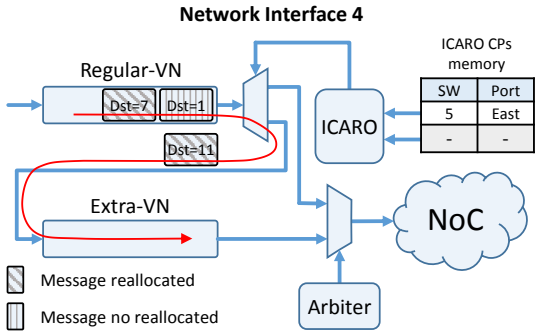


Fig. 7: NI with ICARO for reallocating congested messages.

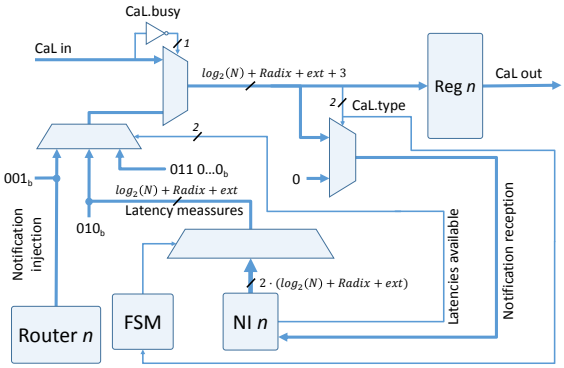


Fig. 8: CaL network register associated logic for regular nodes adapted to DMSD.

congested traffic into the *regular-VNs*. In this way, DMSD will be able to measure the latency of the non-congested traffic (data flowing through the *regular-VNs*). Fig. 7 depicts an example of this process for the NI 4 in a 4x4 2D mesh.

When congestion ends, the conditions leading to detect CPs at routers will not occur anymore. Therefore, all the routers that previously detected CPs will detect this end of congestion and notify this event to the NIs, which will react by removing those CPs from their congestion notification board. Accordingly, since the messages pending to be injected at NIs will not cross any stored CP, those messages will be normally injected through the *regular-VN*.

### B. Delivering latency measurements with the CaL network

In the original DMSD formulation, packets containing the measured end-to-end latency are sent to the controller node via piggybacking [9]. Intense congestion situations, however, may delay the delivery of those packets and the reaction of the PI controller, potentially causing the PI controller to oscillate. On the other hand, ICARO implements the CaL dedicated network, which we modified to support the delivery of those latency values in addition to congestion notifications. By doing this, the metrics necessary to set the frequency properly are guaranteed to timely arrive at destination (with low aggregated overheads, as we show in Sec. III-D).

In a DMSD-based system there are two types of nodes: those that send their latency metrics, and one that receives them. Thus, we implement two slightly different logic blocks to connect to the CaL network. Fig. 8 shows the logic of

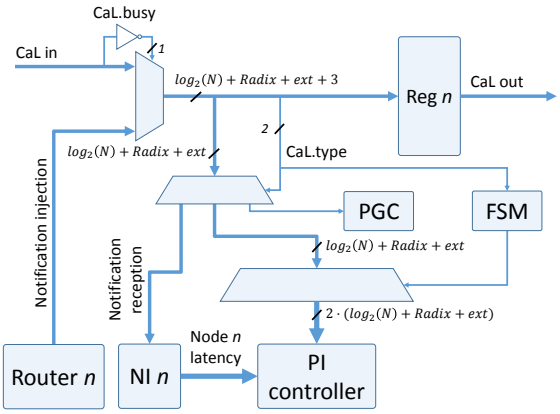


Fig. 9: CaL network register associated logic for the node provided with the PI/DVFS controller adapted to DMSD.

a typical router/NI that sends and receives ICARO notifications and sends DMSD latencies. Note that, despite ICARO notifications are sent in one cycle, we modified the logic to serialize the transmission of 32-bit latencies through the CaL network by extending its links width by (*ext=8-Radix*) bits. Congestion notifications can be seamlessly interleaved with DMSD latency notifications because one bit identifies the type of CaL message. DMSD notifications from different nodes are guaranteed to arrive in order since nodes will send them after a fixed (and different for each node) time offset. Similarly to the sender node in Fig. 8, Fig. 9 shows the logic for the receiver node. This node sends and receives ICARO notifications like any other CaL node, adds its own latency measurements to the received ones, and forwards them to the PI controller.

### C. Power-Gating Extra-VN Buffers

To avoid wasting power when there is no congestion, we implement a mechanism to power-off the extra-VN buffers via a centralized Power-Gating Controller (PGC), which resides in the same node that implements DMSD. All the buffers of an extra-VN (in all NIs as well as in all routers) are powered on/off simultaneously. Power-on is easy: the PGC node snoops the CaL network and when it catches the first congestion notification it broadcasts a power-on message through the CaL network. On the contrary, power-off is not trivial. Snooping the CaL network in search of the "end of congestion" messages is not a valid strategy because there might still be messages in the extra-VN, either in the NIs pending to be injected, in the routers on their way to destination, or both. Therefore, to safely turn off the extra VNs, the PGC must be informed through the CaL network by both all NIs and routers about their detection of a *congestion-free* situation:

*1) Network Interfaces detection:* To make sure that no congested traffic will be injected into the network from a given NI, two conditions must be satisfied. First, the extra-VN buffers must be empty. In addition, the NI congestion notification board must be clean (no new notifications). If both conditions are met, the NI notifies its *congestion-free* status to the PGC with a special message sent through the CaL network.

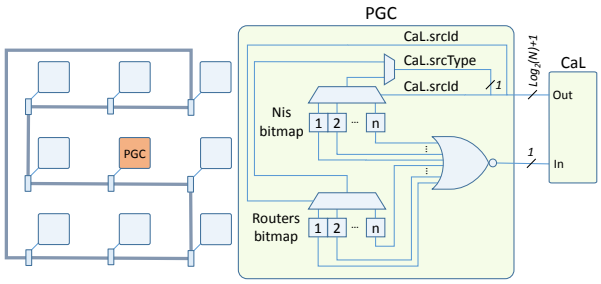


Fig. 10: Power-gating controller.

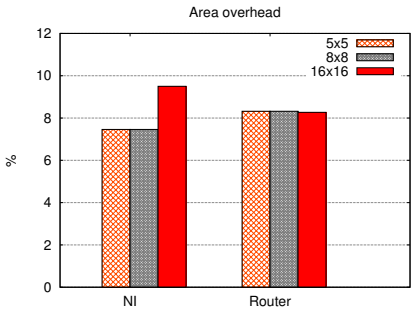


Fig. 11: ICARO+DMSD area overhead of different meshes.

*2) Routers detection:* At routers the mechanism is simpler. Each time the *extra-VN* buffer utilization increases from 0 to 1, the router sends a message to the PGC to inform that is storing congested traffic. On the other hand, when the buffer utilization decreases from 1 to 0, the router sends another message to inform that is *congestion-free*.

The PGC is provided with two $N$-width bitmaps, where $N$ is the number of nodes in the network: one bitmap for the NIs and one for the routers. These bitmaps are updated any time a NI or a router notifies the PGC about its status (0=no congested traffic stored, 1=congested traffic stored). In this way, the PGC has a complete *congestion picture* of the network. When the PGC detects that all NIs and routers are *congestion-free*, it commands to power-off the extra-VN buffers; otherwise, it commands to power-on the buffers. Fig. 10 sketches the PGC bitmaps, the logic to power-on/off the buffers, and its connection to the CaL network. We quantify the advantage of using power-gating in Sec. III-E.

Note that area and power consumed by the PGC are negligible since its implementation only requires $N$-width demultiplexers, 1 $N$-width multiplexer, 1 $N$-width NOR gate and $2N$ registers for a complete mesh.

We are aware that turning on/off all the *extra-VN* buffers is suboptimal since several buffers could not be reached by any congested flow. Because of this, as a future work we plan to implement a new policy to turn the buffers on/off selectively.

### D. Area Overhead Analysis

The bars in Fig. 11 illustrate the area overhead for a NI and a router with support for ICARO, with respect to a baseline implementation (no DMSD, no ICARO)[3]. The results have been obtained after synthesis on our 28-nm technology, in the conditions of Tab. I, except for the mesh size that we let vary. We notice that the overhead is small, less than 10%, even for the case of a large 16×16 mesh.

[3]We obtained overhead results only for ICARO since DMSD and the PG controller overheads are negligible compared to the ICARO's overhead.

### E. Experimental Results

In this section we first report simulation results obtained in the baseline configuration of Tab. I. These results show that our combined DVFS and congestion management strategies can effectively solve the problem outlined in Sec. I that is at the basis of our work. Then, we report results obtained with a sensitivity analysis in which we varied several configuration parameters to check the robustness of our solution. Note that, for our experiments DMSD as well as ICARO are provided with the same amount of VNs in order to compare both solutions with the same amount of resources, providing each VN with the same amount of VCs. However, since DMSD does not require several VNs to work properly and these additional resources may affect negatively to its power consumption we perform an additional experiment comparing against the baseline with only 1 VN.

Figs. 12-14 compare the *DMSD* and the *DMSD+ICARO* cases in terms of latency, frequency, and power, in the baseline scenario. Notice that to properly compare the two cases, the two systems have the same total buffering resources. Note also that, in the case of ICARO, the *extra-VN* is composed of as many VCs as the *regular-VNs*.

Since ICARO effectively separates the background traffic from the hotspot one, DMSD can effectively measure only the latency of the background traffic. Therefore, thanks to the PI controller, DMSD keeps the latency of the background traffic around the 76-ns *latency target*, as shown in Fig. 12. In fact, as Fig. 13 shows, the NoC clock frequency is not influenced anymore by the activation of the hotspot traffic. This, in addition to the use of power-gating, results in a significant improvement of the power consumption, as shown in Fig. 14. When the hotspot is not active (from time 0 $\mu s$ to 300 $\mu s$, and then again after around 380 $\mu s$), the extra-VN buffers are powered-off, resulting in lower power for the *DMSD+ICARO* case. When the hotspot is active, the extra-VN buffers are switched on, hence the power increases. Still, since the clock frequency in the *DMSD+ICARO* case is less than the *DMSD* case, the power consumption is also significantly reduced.

To validate our results under different network configurations, we changed several network parameters: mesh size, router buffers queues size, number of virtual channels, message length, number of hotspots, and hotspot duration. All the cases analyzed are described in Tab. II, in which every case is assigned a label that is used next in the graph keys. As Fig. 15 shows for all the configurations analyzed, in the *DMSD+ICARO* case the background traffic correctly tracks the prescribed target, hence avoiding the excessive power consumption that characterizes the reference *DMSD* case. Note that, since the goal of our approach is to keep the background latency around the latency target, for better understanding, hotspot latencies have been omitted in the graphs. Also note

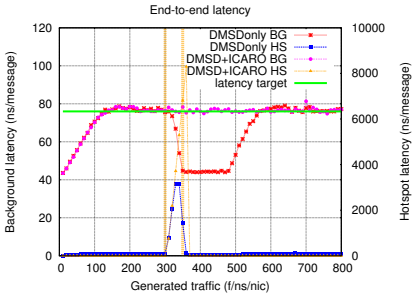


Fig. 12: End-to-end latencies for the background and the hotspot traffic.

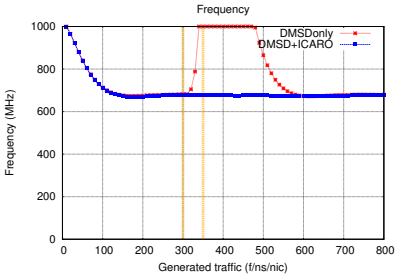


Fig. 13: Frequencies for DMSD and DMSD+ICARO.

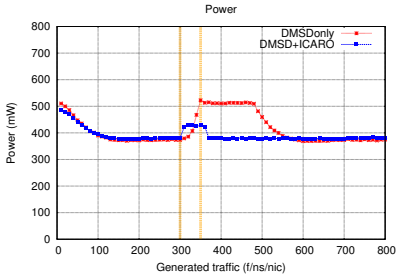


Fig. 14: Power consumption for DMSD and DMSD+ICARO.

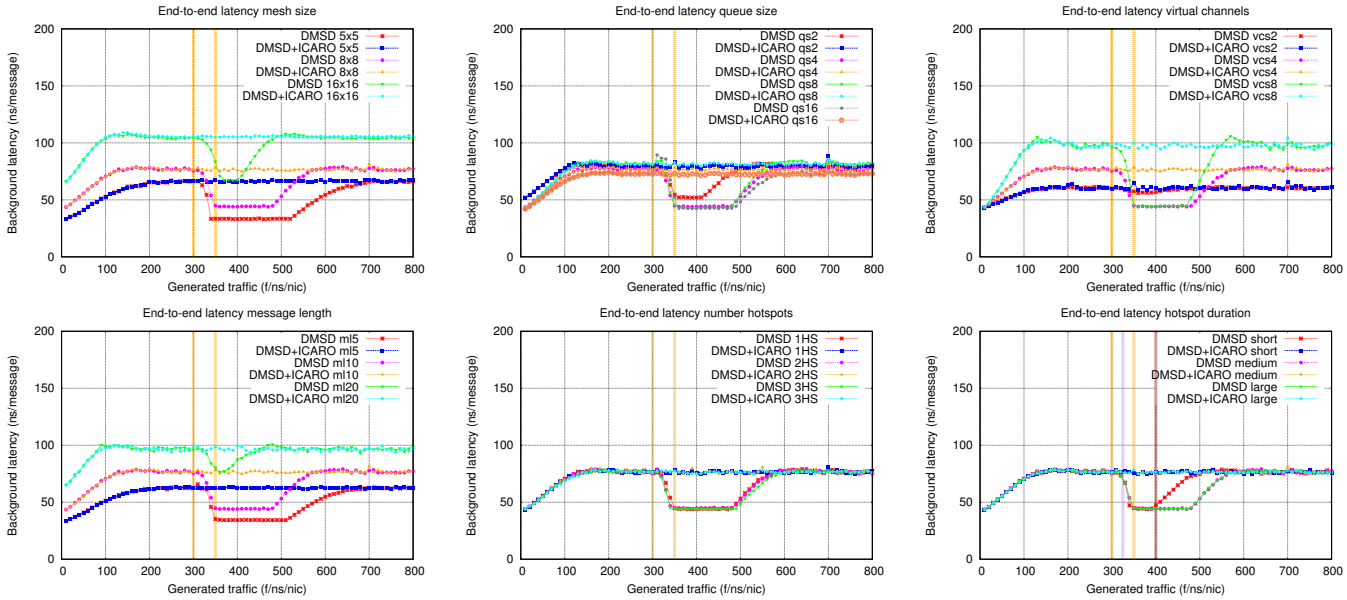


Fig. 15: End-to-end latency for different configuration parameters

TABLE II: Robustness analysis scenarios configuration.

| Scenarios | | | | | | | |
|---|---|---|---|---|---|---|---|
| Label | Mesh Size *(nodes)* | Queue Size *(flits)* | VCs | Msg. Length *(flits)* | Num. HS | HS Dur. *(ns)* | Lat. target *(ns)* |
| Baseline | 8x8 | 4 | 4 | 10 | 1 | 50us | 76 |
| 5x5 | 5x5 | 4 | 4 | 10 | 1 | 50us | 66 |
| 16x16 | 16x16 | 4 | 4 | 10 | 1 | 50us | 105 |
| qs2 | 8x8 | 2 | 4 | 10 | 1 | 50us | 79 |
| qs8 | 8x8 | 8 | 4 | 10 | 1 | 50us | 81 |
| qs16 | 8x8 | 16 | 4 | 10 | 1 | 50us | 72 |
| vcs2 | 8x8 | 4 | 2 | 10 | 1 | 50us | 60 |
| vcs8 | 8x8 | 4 | 8 | 10 | 1 | 50us | 97 |
| ml5 | 8x8 | 4 | 4 | 5 | 1 | 50us | 62 |
| ml20 | 8x8 | 4 | 4 | 20 | 1 | 50us | 96 |
| 2HS | 8x8 | 4 | 4 | 10 | 2 | 50us | 76 |
| 3HS | 8x8 | 4 | 4 | 10 | 3 | 50us | 76 |
| short | 8x8 | 4 | 4 | 10 | 1 | 25us | 76 |
| large | 8x8 | 4 | 4 | 10 | 1 | 100us | 76 |

that the hotspot start/stop time is highlighted with vertical bars and that in the *hotspot duration* graph the three different hotspot ending times are highlighted with different colors. Please note that the *latency target* value for a given scenario depends not only on the saturation point, which is highly correlated with the system configuration, but also on the latency curve gradient. Therefore, in some system configurations

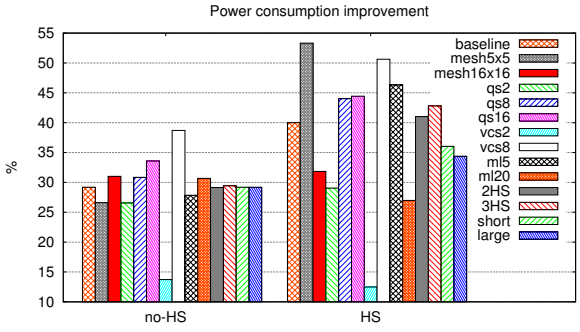


Fig. 16: Power consumption improvement with respect to DMSD for all configurations.

the calculated *latency target* seems not to follow an intuitive progression like in the VCs analysis graph shown in Fig. 15.

Fig. 16 summarizes the improvement of power consumption of the *DMSD+ICARO* case, in all the configurations of Tab. II. Two different improvement values are reported. The first one is due to the *extra-VN* power-gating (*no-HS* in the graph), measured at time 290 $\mu s$ (just before the hotspot activation); the second one corresponds to the power-saving during the

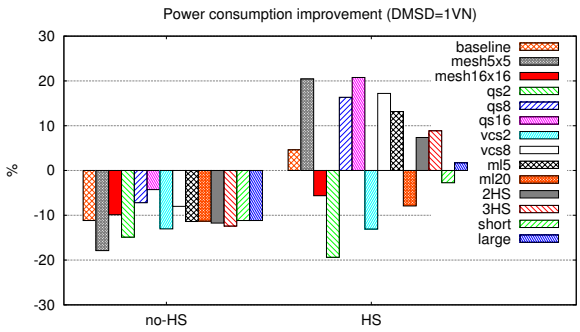


Fig. 17: Power consumption improvement with respect to DMSD (provided with 1VN) for all configurations.

hotspot duration (*HS* in the graph) and is calculated by averaging the power spent from time 300 $\mu s$ to time 600 $\mu s$, since this is the time range in which the hotspots affect any of the cases analyzed. Note that the power overhead due to the additional hardware required by our proposal is already included in the power consumption graphs.

As Fig. 16 shows, for all the cases considered, the combination of DMSD and ICARO leads to a significant power improvement over the DMSD baseline when hotspot is active. When no hotspot is active, by switching the *extra-VN* off we achieve up to 38% power saving and an average of 28%. When congested traffic is detected, ICARO manages this sort of traffic and DMSD tunes the frequency properly saving up to 53% power consumption and 38% on average. In the results obtained when the hotspot is present, we observe a larger variance. This is expected as, for calculating the average, we take values from the same range of time for all cases but the duration of the effects of the hotspots are not the same for those cases, therefore, the weight of those values over the average is not the same.

In the final experiments we analyze our proposal against DMSD provided with 1VN. Unlike ICARO, DMSD does not require several VNs to perform properly, so we performed the same robustness analysis shown above but providing DMSD with 1VN. Nonetheless, as ICARO does not require the *extra-VN* to be provided with several VCs, for this simulations we configured ICARO with the same number of VCs for the *regular-VN* as the DMSD case and only 1VC for the *extra-VN*. The power results in Fig. 17 show that in absence of congestion, ICARO consumes more power than DMSD due to the ICARO logic power consumption. When hotspot is active, however, despite the additional buffers ICARO saves a significant amount of power under the most part of the analysis, achieving up to 20% power saving. Nevertheless, some scenarios present characteristics (amount of resources, message length, etc.) for which DMSD does not overreact to keep the latency under the target, resulting in less power consumption for DMSD. Still, the congestion in those scenarios triggers the ICARO mechanism, causing to switch the extra-VN buffers on, increasing power consumption, and ultimately reducing the power saving compared to the baseline.

## IV. Related Work

Most of the literature focuses on a fine-grain application of DVFS to NoCs, with routers and even links individually powered at different voltages and frequencies [16][17][18][19][20][21]. These works, however, do not consider the overhead of having multiple voltage regulators and PLLs for the various NoC components, not to mention the latency penalty due to multiple clock-domain crossings. We share the view of other authors that consider more practical to have a single voltage and frequency domain for the whole NoC [22][23][24][25].

It is apparent that a fine-grain DVFS approach would lead to better power savings, but the implementation cost would be too high. For these reasons researchers explored a middle ground that we can classify as coarse-grain NoC DVFS, in which either multiple NoC planes (typically two planes) powered at different voltages and/or frequencies are used [26][27], or routers that can individually choose between only two voltages are employed [28]. Our approach can be easily adapted to the case of multiple NoC planes.

In terms of implementation of the DVFS controller, our work has features in common with [24], in which a PI-based DVFS is applied to the NoC and the last-level cache of a Chip Multi-Processor (CMP). A different approach to this problem is proposed in [25], in which the DVFS controller is based on an artificial neural network trained with the help of a PI controller. Differently from these works, we do not restrict our study to the CMP case and analyze the effect of hotspot traffic on the behavior of the PI-based DVFS controller.

Regarding congestion management, most of the solutions in the literature are based on monitoring congestion metrics and using them to make routing decisions. Following this paradigm, RCA [29] uses multiple global metrics collected from the whole network to select at each router the output port which messages are forwarded through. Differently from our approach, RCA collects metrics delivered through the regular network. Thus, if some metrics travel along already congested routes, this may slow down the metrics collection, causing the mechanism to make wrong decisions. Besides, adapting the routes to avoid hotspots may result in moving the location of such hotspots from one place to another. Finally, avoiding hotspots may be impossible if all the flows are bound to the same destination (e.g. the memory controller).

The authors of [30] propose HPRA, a hotspot-formation prediction mechanism. HPRA uses an Artificial Neural Network-based (ANN) hardware that gathers buffer utilization data to predict the formation of hotspots. Then, HPRA classifies the traffic into two classes: hotspot-destined traffic (HSD) and non-hotspot-destined traffic (nonHSD). HSD traffic is throttled at source while the nonHSD traffic is routed avoiding paths containing hotspots routers. However, in the cases in which the ANN fails to predict hotspots, it may redirect traffic to an unpredicted hotspot, causing an even worse degradation of the system performance. Besides, HPRA suffers from the same metrics delivering issue of previously described RCA.

In [31], the authors propose a mechanism to monitor the state of the network in order to select the best path to deliver each packet. To select the best next router for a given packet at each hop, each router in the network must know the best path to follow from the current node to the packet's destination. This requires sending back a special message with the route status information for each message sent from a given node to a given destination. This may cause a waste of network bandwidth and, in presence of several or sudden congestion, it may be affected by the same problem of delayed metrics delivery described before. In addition to this, this mechanism requires that each node keeps a table composed of one entry for each node in the network. This means that the total stored aggregated data in the whole network grows quadratically with the number of nodes, which clearly hampers scalability for large mesh sizes. In contrast, in our case every node only stores a 32-bit latency value, which results in a linear growth with the number of nodes.

## V. Conclusions and Future Work

In this paper we present an integrated approach for saving power while guaranteeing latencies in NoCs under non-stationary traffic patterns. We demonstrate that by integrating a congestion management strategy and a loop-based DVFS controller to tune the frequency for saving power while guaranteeing a latency target, we obtain a power-effective strategy. As our results show, we save up to 53% power compared with the baseline DVFS system in presence of hotspots. In addition to this, we propose a power-gating mechanism to power-off buffers when not needed, resulting in up to 38% power saving in absence of congested traffic. To obtain these results, our approach requires a small area overhead, less than 10%.

As future work we plan to compare the approach reported in this paper with another one that separates traffic classes using physically distinct networks with two different DVFS controllers rather than different virtual networks. In addition to this, as already mentioned, we plan to implement a smarter mechanism to power buffers on/off selectively to achieve best power saving results.

## Acknowledgments

This work was supported by the Spanish Ministerio de Economía y Competitividad (MINECO) and by FEDER funds under Grant TIN2015-66972-C05-1-R and by Ayudas para Primeros Proyectos de Investigación from Universitat Politècnica de València under grant ref. 2370. We also want to thank especially the HiPEAC project that supported the internship during which this work was developed.

## References

[1] L. Benini and G. De Micheli, "Networks on chips: a new soc paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, Jan 2002.

[2] S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb, "The alpha 21364 network architecture," in *Hot Interconnects 9*, 2001, pp. 113–117.

[3] T. Corp., "Tilera tile multicore processors," Available at http://www.tilera.com/products/processors/TILE-Gx_Family.

[4] Kalray. (2014) Kalray, mppa-256 bostan. [Online]. Available: http://www.kalrayinc.com/IMG/pdf/FLYER_MPPA_MANYCORE.pdf

[5] P. Macken, M. Degrauwe, M. Van Paemel, and H. Oguey, "A voltage reduction technique for digital systems," in *37th IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 1990, pp. 238–239.

[6] R. Kumar, D. Tullsen, N. Jouppi, and P. Ranganathan, "Heterogeneous chip multiprocessors," *Computer*, vol. 38, no. 11, pp. 32–38, Nov 2005.

[7] A. Bakhoda, J. Kim, and T. Aamodt, "Throughput-effective on-chip networks for manycore accelerators," in *43rd IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec 2010, pp. 421–432.

[8] T. Sherwood, E. Perelman, G. Hamerly, S. Sair, and B. Calder, "Discovering and exploiting program phases," *Micro, IEEE*, vol. 23, no. 6, pp. 84–93, Nov 2003.

[9] M. R. Casu and P. Giaccone, "Rate-based vs delay-based control for dvfs in noc," in *Proc. DATE*, 2015, pp. 1096–1101.

[10] R. Hou *et al.*, "Efficient data streaming with on-chip accelerators: Opportunities and challenges," in *Proc. HPCA*, 2011, pp. 312–320.

[11] A. Kahng, B. Lin, and S. Nath, "Orion3.0: A comprehensive noc router estimation tool," *Embedded Systems Letters, IEEE*, vol. 7, no. 2, pp. 41–45, June 2015.

[12] J. Escamilla, J. Flich, and P. Garcia, "Icaro: Congestion isolation in networks-on-chip," in *Proc. NoCS*, 2014, pp. 159–166.

[13] D. Lackey *et al.*, "Managing power and performance for system-on-chip designs using voltage islands," in *Proc. ICCAD*, 2002, pp. 195–202.

[14] U. Ogras, R. Marculescu, D. Marculescu, and E. G. Jung, "Design and management of voltage-frequency island partitioned networks-on-chip," *IEEE Trans. VLSI Syst.*, vol. 17, no. 3, pp. 330–341, March 2009.

[15] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Towards the ideal interconnection fabric," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, pp. 150–161, Jun. 2007.

[16] A. K. Mishra *et al.*, "A case for dynamic frequency tuning in on-chip networks," in *Proc. MICRO-42*, 2009, pp. 292–303.

[17] L. Guang, E. Nigussie, L. Koskinen, and H. Tenhunen, "Autonomous DVFS on supply islands for energy-constrained NoC communication," in *Proc. ARCS 2009*, ser. Lect. Notes Comput. Sc., 2009, vol. 5455, pp. 183–194.

[18] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," in *Proc. HPCA*, 2003, pp. 123–124.

[19] J. Zhan *et al.*, "Optimizing the NoC slack through voltage and frequency scaling in hard real-time embedded systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 11, pp. 1632–1643, Nov. 2014.

[20] R. Hesse and N. E. Jerger, "Improving DVFS in NoCs with coherence prediction," in *Proc. NOCS*, 2015, pp. 24:1–24:8.

[21] X. Wang *et al.*, "Fine-grained runtime power budgeting for networks-on-chip," in *Proc. ASPDAC*, 2015, pp. 160–165.

[22] P. Salihundam *et al.*, "A 2 Tb/s 6x4 mesh network for a single-chip cloud computer with DVFS in 45 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 46, no. 4, pp. 757–766, Apr. 2011.

[23] X. Chen *et al.*, "In-network monitoring and control policy for DVFS of CMP networks-on-chip and last level caches," *ACM Trans. on Design Automation of Electronic Systems*, vol. 18, no. 4, pp. 1–21, Oct. 2013.

[24] ——, "Dynamic voltage and frequency scaling for shared resources in multicore processor designs," in *Proc. DAC*, 2013, pp. 114:1–114:7.

[25] J.-Y. Won, X. Chen, P. Gratz, J. Hu, and V. Soteriou, "Up by their bootstraps: Online learning in artificial neural networks for cmp uncore power management," in *Proc. HPCA*, 2014, pp. 308–319.

[26] A. Bianco, P. Giaccone, M. R. Casu, and N. Li, "Exploiting space diversity and dynamic voltage frequency scaling in multiplane network-on-chips," in *Proc. GLOBECOM*. IEEE, 2012, pp. 3080–3085.

[27] J. Henkel *et al.*, "Dark silicon: From computation to communication," in *Proc. NOCS*, 2015, pp. 23:1–23:8.

[28] M. K. Yadav, M. R. Casu, and M. Zamboni, "LAURA-NoC: Local automatic rate adjustment in network-on-chips with a simple DVFS," *IEEE Trans. Circuits Syst. II*, vol. 60, no. 10, pp. 647–651, Oct. 2013.

[29] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *Proc. HPCA*, 2008, pp. 203–214.

[30] E. Kakoulli, V. Soteriou, and T. Theocharides, "Hpra: A pro-active hotspot-preventive high-performance routing algorithm for networks-on-chips," in *Proc. ICCD*, 2012, pp. 249–255.

[31] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Q-learning based congestion-aware routing algorithm for on-chip network," in *Proc. NESEA*, 2011, pp. 1–7.