BlackOut: Enabling fine-grained power gating of buffers in Network-on-Chip routers

Davide Zoni[a,*], Andrea Canidio[a], William Fornaciari[a], Panayiotis Englezakis[b], Chrysostomos Nicopoulos[b], Yiannakis Sazeides[c]

[a] *Politecnico di Milano - DEIB, 20133 Milan, ITALY*
[b] *University of Cyprus - Department of Electrical and Computer Engineering, 1678 Nicosia, CYPRUS*
[c] *University of Cyprus - Department of Computer Science, 1678 Nicosia, CYPRUS*

Abstract

The Network-on-Chip (NoC) router buffers play an instrumental role in the performance of both the interconnection fabric and the entire multi-/many-core system. Nevertheless, the buffers also constitute the major leakage power consumers in NoC implementations. Traditionally, they are designed to accommodate worst-case traffic scenarios, so they tend to remain idle, or under-utilized, for extended periods of time. The under-utilization of these valuable resources is exemplified when one profiles real application workloads; the generated traffic is bursty in nature, whereby high traffic periods are sporadic and infrequent, in general. The mitigation of the leakage power consumption of NoC buffers via power gating has been explored in the literature, both at coarse (router-level) and fine (buffer-level) granularities. However, power gating at the router granularity is suitable only for low and medium traffic conditions, where the routers have enough opportunities to be powered down. Under high traffic, the sleeping potential rapidly diminishes. Moreover, disabling an entire router greatly affects the NoC functionality and the network connectivity. This article presents **BlackOut**, a fine-grained power-gating methodology targeting individual router buffers. The goal is to minimize leakage power consumption, without adversely impacting the system performance. The proposed framework is agnostic of the routing algorithm and the network topology, and it is applicable to any router micro-architecture. Evaluation results obtained using both synthetic traffic patterns and real applications in 64-core systems indicate energy savings of up to 70%, as compared to a baseline NoC, with a near-negligible performance overhead of around 2%. BlackOut is also shown to significantly outperform – by 35%, on average – two current state-of-the-art power-gating solutions, in terms of energy savings.

Keywords: Networks-on-Chip, Static Power, Power Gating, Multi-cores, Low power

1. Introduction

The everlasting demand for more processing power has rendered the multi-/many-core paradigm the de-facto implementation choice in modern CPU architectures. Specifically, multi-core designs are now employed in both embedded and high-performance general-purpose systems, due to their flexibility, performance prowess, and fine-grain control over the chip's power envelope. At the same time, multi-core systems have accentuated the criticality of the on-chip interconnect backbone; traditional bus-based solutions suffer from limited bandwidth capabilities, which cannot cope with increasing numbers of processing cores. Hence, Networks-on-Chip (NoCs) have emerged as the standard interconnect fabric for current and future multi-/many-core implementations. However, the NoC has been shown to consume a significant fraction of the processor's power budget, up to 30% [1]. Hence, proper NoC power containment methodologies have to be considered. In general, typical NoC architectures tend to over-design the implemented resources, in an effort to effectively cope with high traffic conditions without incurring any performance degradation to the overall system.

The way conventional NoCs decide on the buffer depth is based on the "worst-case" scenario of all traffic going through a single virtual channel. Specifically, the buffer depth of each *individual* virtual channel buffer is chosen as to cover the so called credit Round-Trip Time (RTT). Thus, each individual VC buffer is provisioned with this "worst-case" depth, which leads to overall over-provisioning from the perspective of the entire router. Since real traffic is spread among multiple virtual channels, most buffers are almost never filled up.

Furthermore, the amount of *different* virtual channel buffers that must be present in the NoC is typically dictated by the upper-layer cache-coherence protocol, which requires the different message classes (e.g., *request* and *reply* classes) to be separated, to ensure protocol-level deadlock freedom.

Since the NoC faces low traffic for the majority of the time, the over-designed resources remain idle, or under-utilized, thereby wasting energy due to leakage power consumption. The latter is swiftly emerging as a dominant power consumption component in current and future technology nodes.

Extensive prior research has attempted to tackle leakage power consumption in NoCs by exploiting the efficacy of power gating to dynamically switch off unused resources within the

*Corresponding Author
Email addresses: davide.zoni@polimi.it (Davide Zoni), andrea.canidio@mail.polimi.it (Andrea Canidio), william.fornaciari@polimi.it (William Fornaciari), englezakis.panayiotis@ucy.ac.cy (Panayiotis Englezakis), nicopoulos@ucy.ac.cy (Chrysostomos Nicopoulos), yanos@cs.ucy.ac.cy (Yiannakis Sazeides)

NoC. However, power gating introduces a non-negligible delay overhead to wake up the power-gated resources, thus posing a serious threat to overall system performance. Moreover, the traffic traversing the NoC cannot easily be predicted at run-time, which makes predictive wake-up strategies ineffective. The design of effective power-gating methodologies for NoCs – aiming to aggressively reduce leakage power consumption – is usually achieved by means of complex and custom-designed mechanisms. Such architectures exploit additional gather networks to steer information to a centralized power-gating module [2], or they implement complex forwarding networks to timely provide prevailing traffic information to distributed power-gating modules [3].

This work proposes *BlackOut*, a control-theory inspired power-gating methodology that dynamically switches off and on individual NoC buffers. Since the buffers are the leakiest resource within a NoC router, the ability to minimize their static power consumption yields substantial overall reduction in the NoC's total power budget. Most importantly, the power savings are reaped with a negligible impact on overall system performance. The focus of *BlackOut* is on NoC architectures employing a fixed number of statically-allocated VC buffer depths.

The rest of this section is organized in three distinct parts. Section 1.1 details the assumed baseline router, which only represents a use-case for this particular article. Note that *BlackOut* is applicable to any routing algorithm, NoC topology, and router pipeline depth. The rationale behind BlackOut's attempt to minimize the buffers' leakage power consumption is discussed in Section 1.2, while Section 1.3 concisely articulates the novel contributions of this manuscript.

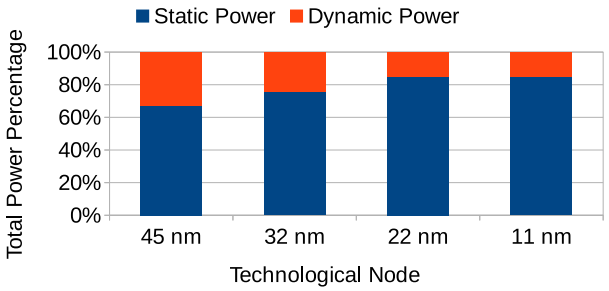
1.1. The assumed baseline NoC router

This article focuses on a wormhole router that supports both Virtual Channels (VCs) and Virtual Networks (VNETs). A generic 4-stage pipelined implementation is assumed with the following stages: (1) Buffer Write/Route Computation (BW/RC), (2) Virtual-channel Allocation (VA), (3) Switch Allocation (SA), and (4) Switch Traversal (ST). A single additional cycle for Link Traversal (LT) is also assumed. As previously mentioned, the proposed BlackOut methodology is not constrained to a specific pipeline depth/micro-architecture; it can be used with any pipeline depth. The proposed solution accounts for the support of VNETs, which are required to enable coherence-protocol support at the NoC level. Traffic within one VNET is isolated from other VNETs, and packets traversing one VNET are not allowed to change VNET in-flight. A packet is split into multiple atomic transmission units, called *flits*. The first flit of each packet is the head flit, which opens up the wormhole. A body flit represents an intermediate flit of the original packet, while the tail flit is unique for each packet and it "closes" the packet and its wormhole. When a head flit arrives at the input port of each router, it has to pass through the 4 aforementioned pipeline stages before traversing the link. First, it is stored in the VC buffer (BW) that has been reserved by the upstream router, and the destined output port is computed (RC). This work assumes the use of Look-ahead RC (LRC) [4], i.e., the destined output port in the downstream router is precomputed in the upstream router. Subsequently, the head flit competes in the VA stage to reserve an idle VC in the selected output port. Note that assigned VCs belong to the set of VCs associated with the VNET of the packet. Once the VA stage succeeds, the head, body, and tail flits compete in packet order for a crossbar switch path (SA). Finally, each winning flit from the SA stage has to traverse the crossbar (ST), and, finally, the output link (LT) before reaching the next (downstream) router. Tail and body flits pass through fewer pipeline stages, since they reuse resources and information reserved by the head flit (i.e., output port and output VC).

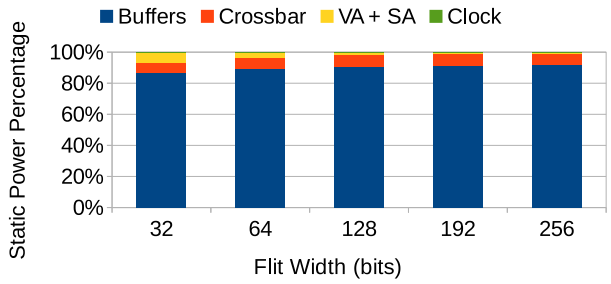
1.2. The rationale behind the BlackOut framework

Figure 1a demonstrates that the leakage power consumption at the 45 nm technology node comprises about 60% of the overall NoC power consumption. As technology scales down to smaller nodes, the static power consumption dominates, reaching 85% of the total power budget. These results were obtained using the DSENT power modeling tool [5], as will be explained in Section 4. Furthermore, the breakdown of a router's static power consumption is depicted in Figure 1b. The top two contributors to the static power consumption are the buffers and the crossbar, but the majority of the static power is clearly consumed by the buffers. Such results are aligned with RTL results presented in [6], which considered a router implemented at 65 nm. This quantitative analysis motivates the focus of BlackOut on minimizing the static power consumption of the NoC buffers.

Static power is typically addressed by exploiting power gating, or by reducing the size of the buffers. At this point of the discussion, it is not yet clear which gating *granularity* would be the most appropriate for this problem, how performance is affected, and how to deal with another crucial issue in NoC design, i.e., the risk of deadlocks. Additionally, the option of reducing the buffer size in the input ports is risky, because smaller buffers incur performance degradation, as shown in Figure 2. The performance analysis was conducted using the *radix* application of the SPLASH-2 benchmark suite [7]. Again, the details of the employed evaluation framework are presented in Section 4. Obviously, a reduction in either the number of VC buffers per VNET (Figure 2a), or a reduction in the VC buffers' depth (Figure 2b) greatly impact the application's execution time. Note that the performance degradation is not so prominent when the number of VCs per VNET is reduced from 4 to 2, but if the VCs are reduced to merely 1 VC/VNET, the performance degradation climbs to 40%. Reductions in the buffer depth also result in performance degradation, as shown in Figure 2b. When the buffers are shrunk from 10 to 8 flit slots, the performance is not affected. Below 4 slots, however, the performance degradation can reach 160%. In both cases, performance initially remains constant but the more aggressive the size reduction becomes, the more impact is observed on overall system performance.

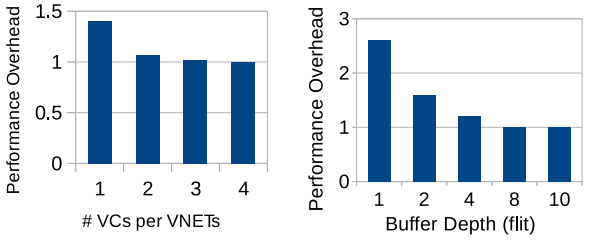


(a) Percentage breakdown of total power consumption vs. technology node, in a NoC router with a 32-bit flit width, while serving traffic injected at a rate of 0.1 flits/cycle.



(b) Percentage breakdown of the *static* power consumption vs. flit width, in a NoC router implemented at the 45 nm technology node.

Figure 1: Analysis of a NoC router's power consumption, using the DSENT power modeling tool [5]. Figure 1a shows the total power consumption breakdown of a NoC router scaling down to smaller technology nodes. Figure 1b presents how the static power consumption breakdown changes when varying the router's flit width.



(a) Performance degradation vs. the number of VCs per VNET.

(b) Performance degradation vs. VC buffer depth.

Figure 2: NoC performance analysis using the *radix* application of the SPLASH-2 benchmark suite [7]. Figure 2a shows how the performance varies with the number of VCs per VNET in each router input port. Figure 2b shows how the performance is affected when changing the VC buffer depth in each router input port.

Clearly, reducing the buffer depth, or the number of VC buffers in each input port, cannot be considered a viable strategy to reduce leakage power consumption, due to the inevitable negative impact on system performance. Consequently, this article proposes the *BlackOut* framework, which is a control-theory inspired methodology exploiting fine-grained power gating of the NoC buffers. BlackOut's ultimate aim is to aggressively reduce the buffers' leakage power consumption, while ensuring negligible performance degradation.

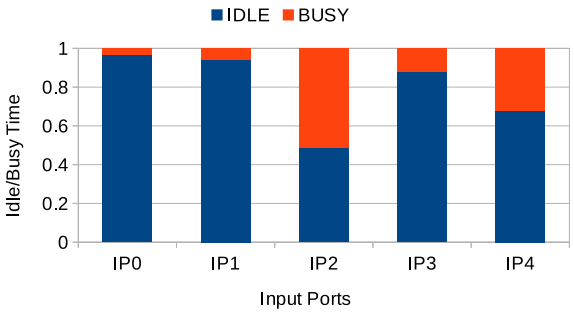
As part of BlackOut's comprehensive evaluation, this work **compares** the proposed methodology **with two current state-of-the-art power-gating mechanisms**, (1) *Power Punch* [3], and (2) the technique in [6]. The *Power Punch* mechanism [3] represents the state-of-the-art in *coarse-grain* power gating (i.e., powering off the entire router), while the mechanism in [6] represents the state-of-the-art in *fine-grain* power-gating at the buffer-level. The experimental results – obtained using both synthetic traffic and real applications – demonstrate that *BlackOut* **substantially outperforms both existing state-of-the-art techniques**. Furthermore, the obtained results indicate that fine-grained power gating at the VC-buffer-level is a much more versatile and effective approach to reducing static power consumption than coarse-grained power gating at the router-level. Figure 3 motivates this assertion, by illustrating the idle periods observed at the port-/router-level, when executing the *ocean* application of the SPLASH-2 benchmark suite [7]. The potential for power-gating of the entire router is much lower than the potential for power-gating of individual VC buffers in each input port. In some cases, this difference in power-gating potential can be as high as 5×. Of course, the static power savings are markedly lower when power-gating a single VC buffer, as opposed to powering off the entire router, but the combined savings from *all* the buffers can exceed the savings reaped from whole-router power gating, as will be demonstrated in this article.

1.3. Novel contributions

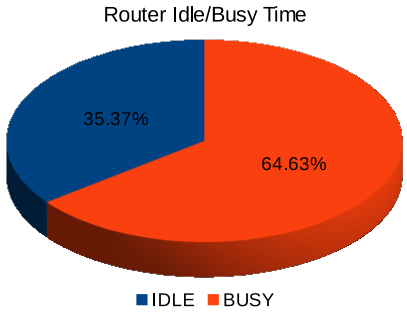
The *BlackOut* mechanism relies on efficient and fine-grained power gating of individual VC buffers within the router's input ports, in order to aggressively reduce leakage power consumption within the NoC. The VC buffers are dynamically switched off and on, based on prevailing traffic conditions, without impacting network (and thereby system) performance. The proposed architecture makes three major contributions to the state-of-the-art:

- **Simple design** – *BlackOut* does not require complex and custom-designed forwarding networks to collect power-gating information. The only communication required is between adjacent router pairs (upstream-downstream), which exchange minimal information through dedicated signals. The mechanism is fully distributed without any centralized coordination; thus, it is scalable to any number of nodes.

- **Operational versatility** – *BlackOut* is not constrained to any particular NoC topology, nor to a specific routing algorithm, or a particular micro-architecture. It is applicable to any VC-based NoC implementation. Moreover, it achieves static power reduction throughout the entire spectrum of injection rates, i.e., from zero-load up to the NoC's saturation point. It outperforms (in terms of energy savings) current state-of-the-art methodologies under all traffic injection rates, by 35%, on average, as will be demonstrated in Section 4. Compared to a baseline NoC without



(a) Average idle vs. busy time at the input-port-level of a NoC router.



(b) Average idle vs. busy time at the router-level.

Figure 3: The observed idle/busy periods when running the *ocean* application of the SPLASH-2 benchmark suite [7], at the input-port-level and at the router-level. As shown in Figure 3a, individual input ports exhibit more opportunities for power gating, since they are idle at least 50% of the time. Power-gating the entire router, however, as shown in Figure 3b, is less opportune, since the idle time (i.e., whole router being empty) is around 35%.

power-gating capabilities, BlackOut reaps per-router energy savings of up to 70%, with a near-negligible performance overhead of around 2%.

- **Optimized micro-architecture** – *BlackOut* relies on a duet of key micro-architectural concepts (discussed in depth in Section 3) that operate synergistically and in unison: (a) *late binding*, and (b) *flow balancing*. The *late binding* technique allows the router to assign a particular buffer to a packet when the packet's head flit arrives at the downstream input port. Since the packet allocation to a buffer is delayed by as much as possible, the buffer allocation process can utilize the available buffer space much more efficiently and optimize the power-gating decisions. The *flow balancing* concept is derived from the observation that traffic variations over short periods of time tend to be small. This implies that the buffer occupancy experiences slight changes over time, thereby allowing for a very effective fine-grained power gating strategy. Overall, *BlackOut* operates on a *sliding window* concept, whereby the window includes all the active (switched on) buffers, and the window may wax and wane by switching on and off additional buffers, respectively. Due to *late binding*, a packet need not be linked to a particular buffer; it only needs to know that a buffer will be available in the downstream router. *Flow balancing* allows for smooth changes to the sliding window size, by adding/removing only a single buffer at a time, without affecting the overall system performance.

The rest of the article is organized as follows: Section 2 discusses related work in the domain of NoC power gating; Section 3 describes the detailed implementation and key attributes of the proposed *BlackOut* architecture; Section 4 presents and analyzes the experimental evaluation results, and Section 5 concludes the article.

2. Related Work

Power gating has been extensively explored within the context of NoCs, in order to reduce the leakage power [8, 3], to improve the reliability of some part of the architecture [9, 10], or both [11]. We categorize the power-gating techniques based on their operational granularity, i.e., either at the router-level (entire router switched off), or at the buffer-level (only individual buffers are switched off).

2.1. Power gating at the router-level granularity

NoRD [12] improves previous power-gating methodologies by focusing on node-router decoupling. By bypassing the input/output ports, the network interface is directly connected to the network, thus communication is allowed even if the router is powered off. To improve the topology, all the bypass links are connected to each other to create a unidirectional ring, which provides communication even if the routers are powered off. Moreover, it is noted that some routers affect performance more than others; thus, to gain advantage of this observation, a distinction between performance-centric and power-centric routers is introduced. The distinction is based on the bypass-link placement, the topology, and the control of wake-up timing.

Router Parking (RP) [8] is a methodology to save power with minimum performance loss, by "parking" routers associated with sleeping cores and pro-actively aggregating traffic to the active routers. A new component, the centralized Fabric Manager (FM), is introduced to the architecture to collect information about the traffic in the NoC. Subsequently, a subset of routers that have their cores powered off are chosen to be parked, and the routing tables of all the routers are updated.

MP3 [13] is a methodology to power-gate routers in Clos networks inducing minimal performance overhead. MP3 maintains a set of routers powered on in such a way that all the processing elements in the network remain connected. Moreover, MP3 selects some components (like input/output ports) in active routers, which are not needed, and shuts them down to save more power.

Catnap [2] splits a single NoC into four smaller physical on-chip networks (sNoCs), which – when combined – have the same bisection bandwidth as the original network. The methodology power-gates each sNoC at the router granularity. If an sNoC gets congested, then the traffic is steered to another sNoC.

In the case that a router in an sNoC is idle for an amount of consecutive cycles, then the router is switched off. Each sNoC has an ID and Catnap steers the traffic to the lower-ID sNoCs first.

DarkNoC [14] proposes a NoC architecture to take advantage of the dark silicon phenomenon in upcoming architectures. DarkNoC organizes the interconnect in multiple physical networks, with each one optimized for a specific frequency. At any given time, only one is active, thus giving flexibility to adjust the power consumption of the NoC.

The Power Punch [3] methodology promises non-blocking power gating in NoC routers. It tries to completely hide the router's wake-up latency of 8 cycles by sending wake-up signals up to 3 hops in advance, adding the capability of hiding a variable number of cycles depending on the router's pipeline. However, it is quite costly to have a wire from every router to every router 3 hops away dedicated to signal wake-ups. Therefore, Power Punch proposes a way to encode wake-up signals to minimize this complexity. The encoding wires are additional to the router-to-router link wires. Power Punch is **the current state-of-the-art in terms of router-level power-gating granularity**. As a result, it is one of the two techniques that will be compared against *BlackOut* later on in Section 4.

All above-mentioned techniques employ power gating at the coarse granularity of an entire router, i.e., they switch off an entire router in the absence of traffic. On the contrary, the proposed *BlackOut* mechanism employs a fine-grained power-gating approach, which is significantly more flexible, since it can selectively switch off individual buffers in each router input port. Whereas a coarse-grained power-gating approach would be unable to switch off a router, even if only a single packet is present, a fine-grained approach would be able to leave the minimum possible number of buffers switched on at any given time.

2.2. Fine-grained power gating

A more flexible approach to power gating is to switch off individual router components at a finer granularity. Of particular interest is the case when such techniques are applied to the router's input buffers, allowing for the shut-down of a portion of the input port.

Ultra-fine-grained run-time power gating [6] discusses a power-gating methodology considering a NoC with only a single VC per VNET. Different parts of the same router can be dynamically power-gated to reduce leakage power. It organizes the router in micro power domains that can be independently power-gated: buffers, VA, crossbar, and output latches. A critical contribution of [6] is the complete RTL design at 65 nm. In particular, the wake-up time for a 128-bit 4-slot buffer is estimated to be 2.8 ns. This result is aligned with [15], where a 4-slot 32-bit buffer is woken up in 1 ns. The wake-up policy relies on look-ahead signals between routers to hide the VC wake-up latency. The authors of [6] also provide various optimization techniques, in order to tackle the head block latency (first-hop wake-up latency). First, in the input ports attached to processing elements, the most utilized VCs remain turned on. Second, VCs in the input ports attached to the caches are pro-actively switched on as soon as an incoming request arrives at the cache node. This work is the most suitable candidate for a comparison against *BlackOut*. It is **the state-of-the-art in buffer-level power gating**, and it is directly comparable to *BlackOut*. Hence, a detailed comparison between the two techniques will be provided in Section 4. The proposed *BlackOut* mechanism relies on a novel *flow balancing* concept, which can dynamically predict the upcoming demand for buffer space. Thus, *BlackOut* can proactively switch on buffers, thereby minimizing the impact on performance. Additionally, the new mechanism uses a dynamic virtual channel buffer remapping technique, which minimizes the number of switch-on and switch-off events.

FlexiBuffer [16] proposes a power gating solution for NoC router buffers considering ultra-fine power domains at a single-buffer-slot granularity. The core of the proposal relies on a novel FIFO buffer design to efficiently switch on and off single buffer slots, without imposing circular buffer utilization, as in traditional FIFO designs. However, due to the implementation overhead, the final solution is to split each buffer in two parts: an always-on part with a few slots, able to handle low traffic demands, and a larger second buffer that is switched on when the number of flits stored in the always-on part of the buffer exceed a certain threshold. A similar solution with double-threshold control has been proposed by Casu et al. in [17], achieving zero performance penalties. On the other hand, *BalckOut* does not require any modifications to the buffer architecture of the NoC router. The policy transparently switches on and off the already existing input buffers. Moreover, FlexiBuffer [16] requires enhancements to the credit-based flow control scheme, whereas *BalckOut* does not.

The Centralized Buffer Router [18] with elastic links and bubble flow control employs a mixed approach between buffer-less and buffered router, depending on the traffic. It uses a buffer-less scheme at low traffic, while a centralized buffer is introduced when the traffic increases. On the contrary, *BlackOut* avoids the complexity of using two different types of buffering. Instead, it uses conventional buffering and simply manages (in an effective manner) how the individual buffers are power-gated, irrespective of the how much traffic is present at any given time.

Panthre [19] exploits routing and topology reconfiguration to optimally use power gating on selected datapath structures in a router. Panthre performs power gating on individual links, which implies that (a) the crossbar connection and the output buffer of the upstream router, and (b) the input port buffers attached to that link downstream, will not be utilized. To avoid/bypass the powered-off segment, an adaptive routing algorithm is used. Unlike Panthre, *BlackOut* does not rely on adaptive routing to avoid power-gated paths. In fact, the NoC routing algorithm is not affected at all, since *BlackOut* wakes up power-gated buffers just-in-time to be used by incoming flits.

3. The BlackOut Methodology

This section describes the details and key attributes of the proposed *BlackOut* mechanism, which is a control-theory

inspired methodology that power-gates individual input VC buffers in NoC routers to reduce the static power consumption. Said methodology is able to aggressively power off the majority of the input buffers in a router, while incurring negligible overhead to the overall system performance.

BlackOut is implemented on top of credit-based flow control, with two main design constraints. First, the changes are minimal and the design is very lightweight. Second, *BlackOut* directly exploits the flow control mechanism, so that it is guaranteed not to steer the system into a deadlock, as long as the baseline system is guaranteed to be deadlock-free.

The *BlackOut* design methodology is split in two parts and encompasses a generic pair of actors: the upstream and the downstream modules. The *physical actuation* is performed by the downstream module, i.e., the router input port where the buffers are situated. Conversely, the upstream module is in charge of the *control-and-command dispatches*, namely signaling to the downstream module to switch off or on a particular buffer. This action is performed by the router's output port, or by the Network Interface Controller (NIC). Therefore, each router implements one instance of the upstream module policy per output port, and one instance of the downstream module policy per input port. Instead, the NIC only requires an implementation of the upstream module policy in its output port, since the NIC is never considered to be a downstream module.

The rest of this section is organized in four parts. Section 3.1 presents a bird's-eye view of the key ideas behind *BlackOut*. An in-depth discussion of the policies considering router-to-router and NIC-to-router *BlackOut* implementations is provided in Section 3.2 and Section 3.3, respectively. Finally, Section 3.4 presents the complete *BlackOut* proposal and includes an example of *BlackOut*'s execution, through the use of a detailed timing diagram.

3.1. Conceptual overview of BlackOut

BlackOut manages the on/off state of buffers by employing a **sliding window**. At any point in time, a subset of the available buffers in the input port are idle, and they can potentially be switched off. Depending on the traffic conditions, the sliding window is incremented, or decremented, by switching on or off a new buffer, respectively. Thus, a window can be fully closed (when all buffers are switched off), or completely open (when all buffers are switched on). *BlackOut* is founded on two different operational pillars, which provide flexibility and great power-saving capabilities with negligible performance overhead: (a) *flow balancing*, and (b) *late binding*.

Without loss of generality, the rest of this section considers an upstream router Output Port (OP) and a downstream router Input Port (IP) pair. The *flow balancing* concept applies to the upstream router output port, and is linked to the inherent balance between the current buffer occupancy in the downstream input port and the one in the near future. In particular, the SA requests in the upstream router – that are routed to the OP – represent the buffers currently in use in the downstream router IP. They passed the VA stage, so a buffer is allocated in the downstream router for each of them. Conversely, the portion of requests in the upstream router that are routed to the OP and are in the VA stage represent the future buffer requirements. The VA requests directed to the OP can be more than the SA requests, thereby implying that the number of active buffers cannot cope with the packet flow. On the other hand, if the VA requests are lower than the SA requests for the considered OP, than the number of active buffers can manage the VA requests in the future without the need to switch on other buffers. In other words, the *flow balancing* concept defines the OP state as the current state, i.e., the active buffers or SA requests, and the current VA requests as the future state. This basic information is used to develop a policy to increase, or decrease, the number of active (on) buffers in the corresponding downstream IP.

The *late binding* technique optimizes the physical allocation of a packet to a buffer in the downstream IP. In a NoC, the virtual and physical allocation of a buffer to a packet happens at different times, since the VA in the upstream router takes place several cycles before the BW in the downstream one. *Late binding* is a mechanism to decouple virtual allocation of the channel from physical binding, eventually re-mapping a virtual channel onto a different physical buffer. This means that the upstream VA exploits the baseline flow control policy, while the head flit of the packet is linked to a particular physical buffer only when it actually arrives at the IP of the downstream router. A buffer is then assigned, depending on the current availability and state (on/off) of the buffers in the IP. It is worth noting that a packet arrives at the downstream IP if the VA in the upstream node is successful. Hence, no deadlock, or lack of resources can result from *late binding*, since no speculation is involved. Only a buffer mapping occurs at the moment of arrival.

3.2. The BlackOut router-to-router model

BlackOut distinguishes between the NIC and the router output ports. In this section, the methodology applied between two routers is described. In the downstream router, the focus is on its input port buffers, while in the upstream router, the focus is on the traffic injector, on the control decisions, and the router's output port.

Figure 4 highlights the modified router architecture to accommodate the *BlackOut* methodology. Router R_0 has 3 packets in its input ports directed to R_1. The red ones are already in the SA stage, while the blue one may be in the BW, or the VA stages. R_0's output channels are marked when there is a packet allocated to them; specifically, the red channels are the allocated ones. The R_0's policy has to decide whether to request a new channel, or to wait until an already-on channel will become free. This decision is signaled to R_1 through the action signal. The packets allocated in R_0 on a certain channel may be physically allocated to a different input buffer in R_1, thanks to the re-mapper, which is placed before R_1's input buffers to guarantee late binding. Moreover, the re-mapper module makes the late binding mechanism transparent to the upstream router. Figure 4 also provides a view of the sliding window, which is clearly visible in R_1's input port, as the rectangle around the set of switched-on input buffers.

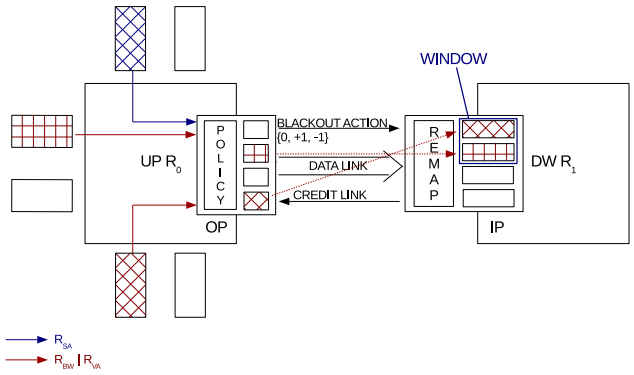


Figure 4: General overview of the *BlackOut* router-to-router architecture. The upstream and downstream modules are depicted, which highlight the differences/additions introduced by the *BlackOut* methodology.

3.2.1. Upstream router

The upstream module is the *control-and-command dispatcher* of *BlackOut*. The methodology works on a per-OP basis. Considering a single OP, the mechanism senses the traffic and, depending on the needs, gives the signal to enlarge or reduce the window, i.e., the number of active (on) buffers.

The upstream router examines the metrics defined below to resize the window of a specific OP. All the required information is reported per VNET, since the traffic between the different VNETs of a network is different, and a buffer can only store packets from a single VNET at a given time.

$R_{BW_{j,i,t}}$ defines the requests in the BW stage directed to OP j for VNET i at time t. $R_{VA_{j,i,t}}$ defines the requests in the VA stage directed to OP j for VNET i at time t. $R_{SA_{j,i,t}}$ defines the requests in the SA stage directed to OP j for VNET i at time t. They represent the characterization of the traffic the output port is experiencing. The requests in the BW and VA stages can be viewed as incoming traffic, while those in SA as active traffic.

Moreover, $W_{OP_{j,i,t,used}}$ defines the number of VCs used in OP j for VNET i at time t, $W_{OP_{j,i,t,idle}}$ defines the number of VCs in idle state in OP j for VNET i at time t, and $W_{OP_{j,i,t,NAVCA}}$ is the number of VCs allocated in Non-Atomic Virtual-Channel Allocation (NAVCA) mode in OP j for VNET i at time t. Finally

$$W_{OP_{j,i,t,usable}} := W_{OP_{j,i,t,idle}} + W_{OP_{j,i,t,NAVCA}} \quad (1)$$

captures the number of VCs usable in OP j for VNET i at time t, which are the VC buffers in idle or used channels that can be allocated in NAVCA mode. The policy developed has two stages. First, a local decision is taken concerning a single VNET. Then, a global policy takes all the local policy results to impose a global decision for the OP, taking into account all the VNETs.

Algorithm 1 shows that a local decision to shut down a channel is taken ($R_{j,i,t+1} \leftarrow -1$) when the number of usable VCs in OP j for VNET i is greater than 0, and the incoming requests are at most equal to the active ones. This means that the traffic is decreasing, or it is constant, and some active channels are unused. Conversely, a local decision to turn on a channel is taken ($R_{j,i,t+1} \leftarrow +1$) when the number of usable VCs is at most 0, and the incoming requests are greater than the active ones. This means the traffic is increasing and there are not enough channels to handle it. Otherwise, a local decision to maintain the same number of channels is taken ($R_{j,i,t+1} \leftarrow 0$). The global decision policy for a specific output port is the mechanism that merges the local decisions from all the VNETs into a single one, as shown in Algorithm 2.

Algorithm 1 Local decision for output port j and VNET i at time t+1

$R_{j,i,t}$:= Local decision in OP j for VNET i at time t

if $W_{OP_{j,i,usable}} > 0$ **then**
  if $R_{BW_{j,i,t}} + R_{VA_{j,i,t}} \leq R_{SA_{j,i,t}}$ **then**
    $R_{j,i,t+1} \leftarrow -1$
  else
    $R_{j,i,t+1} \leftarrow 0$
  end if
else
  if $R_{BW_{j,i,t}} + R_{VA_{j,i,t}} > R_{SA_{j,i,t}}$ **then**
    $R_{j,i,t+1} \leftarrow +1$
  else
    $R_{j,i,t+1} \leftarrow 0$
  end if
end if

Algorithm 2 Global decision for OP j at time t+1

$W_{total_{j,t}}$:= Window size in OP j at time t
$R_{j,t}$:= Global decision in OP j at time t

$W_{total_{j,t+1}} = W_{total_{j,t}} + R_{j,t+1}$

if $\left|(R_{j,i,t+1} == +1 \right.$ **then**
  $R_{j,t+1} = +1$
else
  if $\left\|\left[(R_{j,i,t+1} == -1) \wedge (W_{OP_{j,i,idle}} > 0) \right]\right.$ **then**
    $R_{j,t+1} = -1$
  else
    $R_{j,t+1} = 0$
  end if
end if

The global decision depends on the local decisions, and the focus of the local decisions is to maintain system performance. Hence, whenever a local decision requests a switch-on, the global decision will request a buffer switch-on, too. Conversely, the global decision requires to switch off a buffer if no switch-on signals are coming from any of the local decisions, at least one local decision requires a switch-off, and there are idle buffers.

The upstream router is in charge of selecting the target VC, based on the global decision that has been taken. This solves the scenario where multiple local decisions request to switch on a new buffer. For example, Algorithm 3 describes how to choose a target channel to power on. The idea is to select a switched-off channel from one of the VNETs that requested a switch-on.

Algorithm 3 Channel selection for power-on in an OP

```
1: for all vnet in outport do
2:     if R_{outport,vnet,t+1} == +1 then
3:         for all vc in vnet do
4:             if vc == OFF then
5:                 return vc
6:             end if
7:         end for
8:     end if
9: end for
10: return −1
```

Algorithm 4 Channel selection for power-off in an OP

```
1: targetVC ← −1
2: for all vnet in outport do
3:     if R_{outport,vnet,t+1} == −1 then
4:         for all vc in vnet do
5:             if vc == ON ∨ vc == IDLE then
6:                 targetVC = vc
7:             end if
8:         end for
9:     end if
10: end for
11: return targetVc
```

Similarly, Algorithm 4 shows how the upstream router selects a target channel to shut down, because a global decision for a switch-off has been taken. A powered-on idle channel is selected from one of the VNETs that requested a switch-off. The upstream router sends the decision to the downstream router (see Section 3.2.2 for a detailed discussion). The methodology always keeps a buffer switched on in each output port, under the router-to-router policy. It has been observed that this decision greatly reduces the performance penalty that a power-gating scheme may incur.

3.2.2. Downstream router

The downstream router is the physical actuator of *BlackOut*. This means that the downstream router has to perform the power-on and power-off of the physical buffers, based on the signals received from the upstream router. It is also responsible for the late binding mechanism, in order to optimize the buffer on/off switching (i.e., power-gating) activity.

The re-mapper, which is able to allocate an upstream VC to a different physical buffer is introduced to provide late binding, as shown in Figure 4. A re-mapping table is needed to record which VC is mapped to which physical buffer. The binding starts when the first flit of the packet arrives at the downstream router for the BW stage, and it ends only when the physical buffer is idle, and a power-off request has been received. The upstream router directly targets a particular VC to power off, and it needs to communicate the targeted VC to the downstream router to remove the mapping from the re-map table.

When a flit – directed to an unmapped VC – arrives, the re-mapping policy finds an unmapped switched-on buffer. This buffer is then selected and mapped to the VC, so that any packets using that particular VC will be directed to the VC's newly-mapped buffer. The re-mapping policy steers the traffic to the low-ID buffers, leaving the others almost idle. The physical buffer is now independent from the VC assigned to each VNET, due to the re-mapper. The re-mapper can always find a physical buffer to map to a VC, by construction.

Whenever a request for a new buffer is received, the policy needs to target a new physical buffer to power on. The target physical buffer (to be switched on) is decided by iterating over all the VC buffers in the input port, and by selecting the first (lowest ID) powered-down buffer. It is always ensured that whenever a new channel is requested, a physical buffer will be available, because, as mentioned in Section 3.2, the policy always functions within the capabilities of the router.

The same reasoning is applied when the downstream router receives a request to switch off a channel. The power-gating actuator needs to target a specific physical buffer among the ones that are already switched on, or that are switching on. The buffer switch-on process is, in fact, not instantaneous, i.e., it requires a certain number of clock cycles before it completes. Thus, a buffer that is marked as "switching on" will become ON in the near future (a few cycles later). Consequently, the *BlackOut* technique considers buffers that are in this transitional mode (i.e., in the process of switching on) as being active and eligible for power gating. *BlackOut* prioritizes the buffers that are in the process of switching on to be power-gated, in order to increase the power savings. Again, the VC buffers with the lowest ID are chosen (either the ones that are in the switching-on phase, or the ones that are on and operating).

3.3. The BlackOut NIC-to-router model

This section describes the *BlackOut* operational model between a NIC and a router, where the NIC is considered to be the upstream module and the router represents the downstream module.

The same policies are used to generate the on/off signals, but other metrics need to be defined for the policy, because of the differences between the NIC and the router. In contrast to the router-to-router policy, where a single VC is always left on, in the NIC-to-router policy no VC is left switched on if the NIC is attached to a memory controller, or an L2 cache bank. This decision was taken because, according to the examined benchmarks, the traffic to the input ports connected to those modules is negligible.

Figure 5 highlights the modified architecture for the NIC and the router, considering the NIC-to-router model. As an example, let us consider an upstream module NIC_0 and a downstream module R_0. Since the downstream module is identical to the one presented in Section 3.2, we focus here on the upstream NIC. NIC_0 has three messages in its message queues: the red ones are in the Link Allocation (LA) stage (the NIC equivalent to a router's SA stage), while the blue one is in the VA stage. NIC_0's output channels are marked to distinguish whether or not their allocation is possible; only the red ones are available here. NIC_0 has to decide whether to request a new channel, or

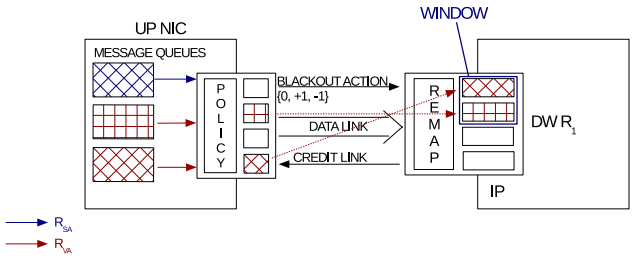


Figure 5: General overview of the *BlackOut* NIC-to-router architecture. An upstream NIC and a downstream router are shown, which highlight the differences/additions introduced by the *BlackOut* methodology.

to wait for an already-on channel to become free. The decision is passed on to R_0 through the action signal.

Note that the minimum number of queues (i.e., VC buffers) in each input port is imposed by the cache coherence protocol used. Directory-based cache coherence protocols in multi-core systems categorize all cache traffic into certain *message classes* [20, 21]. For example, all memory requests sent from a processing core to a cache controller belong to the *request message class*. Similarly, the memory contents returned from the cache to the core (e.g., a cacheline) belong to the *response message class*. In order to avoid protocol-level deadlocks, these cache coherence protocols **require** that the various message classes are isolated and have their own dedicated buffers [20]. The reason is simple: if the request and response traffic shared the same buffers, there could be a case where all the buffers could be filled with multiple outstanding memory *requests*, with no buffers left for the *response* traffic. Therefore, to provide isolation between the coherence protocol's message classes, modern NoCs employ **different virtual channels for each message class** (with dedicated buffers for each virtual channel). Consequently, the minimum number of VC buffers in each input port is imposed by the coherence protocol.

The NIC-to-router upstream policy is the same as the router-to-router upstream one. However, the architectural differences between the router and the NIC require the definition of new metrics. The NIC gets the messages from the message queues, it breaks them up in to flits, and allocates the resulting flits to an output VC. Then, it arbitrates the output requests competing for the link, and, finally, sends the chosen flit.

$R_{VA_{i,t}}$ is defined as the number of requests in the message queue for VNET i at time t. $R_{SA_{i,t}}$ represents the number of requests ready for LA for VNET i at time t. The messages in the queues represent new incoming traffic, while the requests in the LA stage represent the active traffic.

Moreover, $W_{i,t,used}$ is defined as the number of channels used for VNET i at time t, while $W_{i,idle}$ represents the number of channels in idle state for VNET i at time t. Finally, $W_{i,t,usable} := W_{i,t,idle}$represents the number of channels usable in the NIC for VNET i at time t. The local decision policy is described in Algorithm 5.

If there are idle channels, and the requests in the message queue are fewer than those in LA, then the traffic is decreasing, and the algorithm triggers the local decision to switch off a channel. Note that in the router's local decision policy, a switch-off choice is made even if the traffic is constant. A channel can also be switched off when there are no requests in the NIC, i.e., there is no traffic in it. On the other hand, the local decision to switch on is taken if there are not enough usable channels, and the number of the incoming requests is greater than, or equal to, the number of the active ones; namely, the traffic is increasing. The $\neg(R_{VA_{i,t}} == 0 \wedge R_{SA_{i,t}} == 0)$ boolean expression is added to the switch-on condition to prevent policy oscillations in the absence of traffic (to be explained below). Again, if none of the above conditions is met, the local decision is to maintain the same number of active channels.

Algorithm 5 Local decision in NIC for VNET i at time t+1

1: $R_{i,t}$:= Local decision in NIC for VNET i at time t
2:
3: **if** $W_{i,usable} > 0$ **then**
4: **if** $(R_{VA_{i,t}} < R_{SA_{i,t}}) \vee (R_{VA_{i,t}} == 0 \wedge R_{SA_{i,t}} == 0)$ **then**
5: $R_{i,t+1} \leftarrow -1$
6: **else**
7: $R_{i,t+1} \leftarrow 0$
8: **end if**
9: **else**
10: **if** $(R_{VA_{i,t}} \geq R_{SA_{i,t}}) \wedge \neg(R_{VA_{i,t}} == 0 \wedge R_{SA_{i,t}} == 0)$ **then**
11: $R_{i,t+1} \leftarrow +1$
12: **else**
13: $R_{i,t+1} \leftarrow 0$
14: **end if**
15: **end if**

Regarding the possibility of policy oscillations in the absence of traffic, the problem would occur in the corner case when there are zero VA and SA requests. The second condition on line 4 of Algorithm 5 (zero VA and SA requests, i.e., absence of traffic) will request a VC buffer to be switched *off*. In the next cycle, the first condition of line 10 of Algorithm 5 (ignore the second condition for now) would request a new VC buffer to be switched *on*, because the number of VA requests equals the number of SA requests (even if there is no traffic). In the following cycle, line 4 of Algorithm 5 would switch *off* that buffer for the reason mentioned above (zero VA and SA requests, i.e., absence of traffic), and so on. Hence, in order to avoid this oscillatory behavior, we have *added the second condition* on line 10 of Algorithm 5, i.e., $\neg(R_{VA_{i,t}} == 0 \wedge R_{SA_{i,t}} == 0)$, which prevents the BlackOut policy from switching *on* a VC buffer when the numbers of VA and SA requests are both equal to 0.

3.4. Overall operation and a corner-case scenario

This section presents an example of the entire operation of *BlackOut*, which combines all the models and algorithms presented in the previous sub-sections. Additionally, a specific corner-case scenario is presented, which could *potentially* impact the NoC performance. However, the corner case is extremely infrequent and – in all evaluated configurations presented in Section 4 – it does not affect the overall system performance.

Figure 6 depicts the detailed execution of the *BlackOut* mechanism, assuming a four-flit packet is traversing a pair of

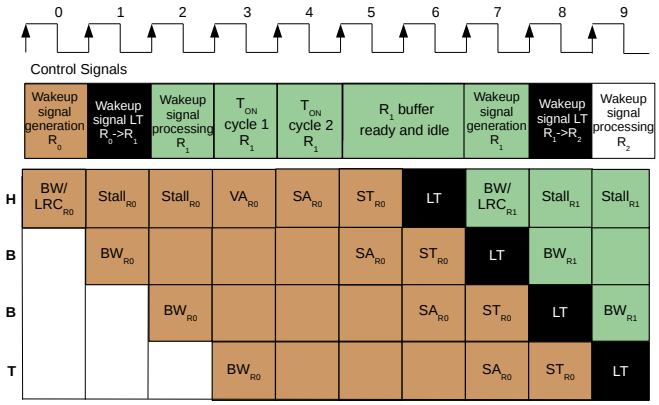


Figure 6: A cycle-by-cycle example of the operation of the *BlackOut* mechanism, assuming a four-flit packet is traversing a pair of upstream/downstream routers. The upstream router is R_0, while the downstream router is R_1. Router R_2 is further downstream.

upstream/downstream routers along its path to its final destination. The various flit types (i.e., head, body, and tail) are denoted in the figure by the letters 'H,' 'B,' and 'T,' respectively. For each flit of the packet under investigation, Figure 6 depicts two consecutive router pipelines, i.e., one for the upstream router (R_0) and one for the downstream router (R_1). The top part of the figure (below the clock signal) illustrates the control signals and actions taken by the *BlackOut* mechanism in the upstream and downstream routers. Figure 6 assumes a T_{ON} latency (i.e., the number of cycles required to switch on a VC buffer) equal to 2 cycles, and that no buffers are active and idle in cycle 0.

A new head flit is detected by the policy in the Buffer Write (BW/LRC) stage in cycle 0 in the upstream router (R_0). In the same stage, since the incoming requests are greater than the active ones, the signal to power-on is generated. In the upstream module, the VA is stalled for T_{ON} cycles, so a delay in the pipeline is observed. The power-on signal is received by the downstream router (R_1) two cycles later (in cycle 2); one cycle is spent to traverse the link (cycle 1), and another one (cycle 2) to be received and processed by the downstream router itself. When the downstream router processes the received signal (end of cycle 2), it targets a VC buffer to power on, which will be available after T_{ON} cycles.

It is worth noticing that by stalling the upstream router's VA stage by two cycles, i.e., the T_{ON} latency, the downstream input buffer is ready two cycles in advance of the actual time the flit will be stored in it. This gap is due to the upstream router's pipeline traversal. Thus, a total of two cycles of the T_{ON} latency can be hidden when a four-stage pipeline is used. In other words, the stall in the VA stage of the upstream router was unnecessary in this example, since the two cycles of the T_{ON} latency could have been *completely hidden* by the upstream router's pipeline traversal. However, if we considered a T_{ON} latency of 4 cycles with the current configuration, we would need a 2-cycle stall in the upstream router's VA stage to ensure that the buffer in the downstream router would be ready to accept the flit upon its arrival. Hence, the crucial observation here is that a stall is introduced in the VA stage of the upstream router only if (a) a new buffer has to be switched on in the downstream router, and (b) if the T_{ON} latency is larger than the pipeline-maskable delay, i.e., two cycles in our example configuration. In a similar vein, a three-stage pipeline allows for a single cycle of the T_{ON} latency to be hidden, and the full T_{ON} latency has to be paid in a 2-stage pipeline.

Considering the entire *BlackOut* implementation, there is a single corner-case scenario that may potentially affect performance: the so called *serialization effect*. Performance degradation may be observed under this scenario, due to the conservative behavior of the policy in switching on and off the buffers. In particular, when the number of VA requests and SA requests are equal, *BlackOut* prevents an additional channel to be switched on, thus incurring a serialization problem imposed by the flow balancing concept. However, this conservatism minimizes the oscillations in the buffers' switching activity (i.e., switching on/off).

Figure 7 depicts the problematic serialization scenario. The P_3 flit in R_0 is blocked, even if its path is free. This happens because the P_2 flit is in R_1, and it happens to be blocked by another flit. So, it keeps the channel in R_0 active, while waiting for credits to arrive. Flit P_3 is blocked, due to the flow balancing assumption in *BlackOut*, which prevents a channel switch-on if the numbers of incoming and active requests are equal. Additionally, even P_2 suffers from the same type of problem, since, in R_1, the numbers of active requests and incoming ones are equal, due to flit P_1 residing in R_2. Nevertheless, our experiments indicate that this serialization effect is rarely observed, so its potential impact on performance is not noticeable.

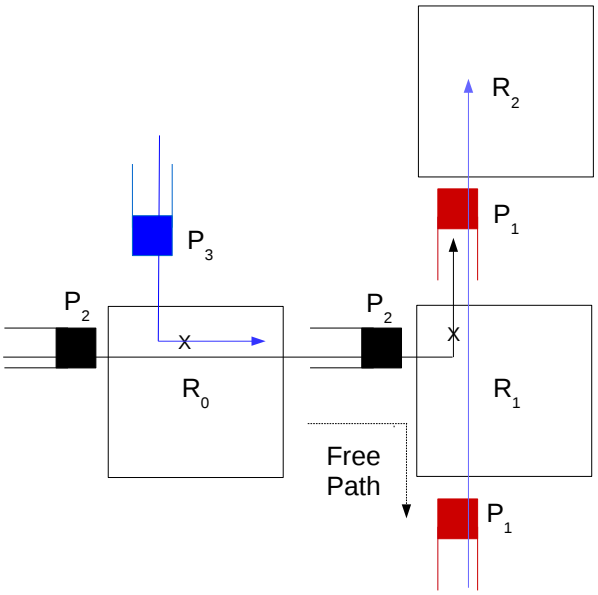


Figure 7: An example of the so called *serialization effect*, which is a problematic corner-case scenario that may potentially impact performance when using *BlackOut*. However, this scenario is rarely observed, as indicated by our experiments.

4. Evaluation and results

The assessment of the *BlackOut* methodology of Section 3 considers performance/energy savings, and the hardware over-

Processor Core	1 GHz, Out-of-Order Alpha Core
L1I Cache	2-way, 32 kB
L1D Cache	2-way, 32 kB
L2 Cache-	8-way, 256 kB per bank
Coherence Prot.	MOESI (3-VNET protocol)
Router	4-stage Wormhole
	32-bit Link Width
	Buffer Depth 4 Flits
	2 Virtual Channels (VCs) per VNET
	3 VNETS (Garnet Network [22])
Topology	2D-mesh / 8×8 (1 core/tile)
Routing Alg.	Deterministic XY
Technology	45 nm at 1.0 V
Synthetic	Uniform Random, Tornado,
Traffic Patterns	and Transpose
Real Traffic	SPLASH-2 benchmarks [7]

Table 1: Experimental setup: salient parameters pertaining to the processor, router micro-architecture, assumed technology, and employed traffic patterns.

head of the new design. This evaluation is distributed over the following four sub-sections. Section 4.1 describes the simulation setup and the two state-of-the-art reference designs used for comparisons against the proposed architecture. Performance and energy simulations considering both synthetic traffic and real applications are then presented in Section 4.2 and Section 4.3, respectively. Finally, the hardware implementation analysis of *BlackOut* is presented in Section 4.4.

4.1. Experimental setup

BlackOut has been fully implemented and integrated into an enhanced version [23, 24] of the *gem5* cycle-accurate simulator [25], extended in [23, 24] to also include a cycle-accurate model of power gating in the NoC. DSENT [5] is used to extract power data of the simulated NoC architectures. Furthermore, two different state-of-the-art proposals [3, 6] – both discussed extensively in Section 2 – have been implemented and integrated in the simulation framework to serve as comparison points (in terms of performance and reaped energy savings) against *BlackOut*. In particular, *Power Punch* [3] implements power gating at the router-level granularity, while the techniques in [6] facilitate power gating at the buffer-level granularity. Thus, a complete evaluation is provided against the current best-in-class solutions that leverage power gating at the router-level (coarse-grain) and the buffer-level (fine-grain).

For each of the considered methodologies, the additional energy required to wake up an entire router is obtained using the analytical models proposed in [2], and the energy overhead to wake up a single buffer is computed by appropriately scaling the energy required to wake up the entire router. The wake-up time delay is assumed to range between 1 and 4 cycles for a single buffer, based on the RTL results presented in [6]. For the router-level comparisons, the wake-up latency of an entire router is assumed to be 8 cycles, as reported in [3].

Although *BlackOut* is not constrained to any NoC topology or routing algorithm, the assessment is performed – without loss of generality – on 64-core 2D-mesh tiled architectures using XY routing, whose detailed parameters are reported in Table 1. Each tile is composed of an out-of-order core with private, split L1, and shared L2 cache. The L2 is physically split into 64 banks, i.e., one per each tile. Moreover, each tile has a router that allows the connection with the rest of the architecture. Note that the core and the private L1 are substituted with a traffic generator when synthetic traffic is considered.

Three different synthetic traffic patterns are considered, i.e., uniform random, tornado, and transpose, as well as nine SPLASH-2 [7] benchmarks configured to allow a maximum of 64 parallel threads.

Regarding the power-gating actuator, the wake-up time represents a critical parameter that can outweigh any benefit of the methodology. To assess the criticality of this wake-up latency, *BlackOut* is evaluated considering three different latencies to wake up a single buffer (T_{on}), i.e., 1, 2, and 4 cycles. This means that, once the wake-up command reaches the downstream router, the selected buffer is ready to be used T_{on} cycles later, where the current cycle is not considered to be part of T_{on}. An instantaneous switch-off is assumed for all the considered methodologies. Thus, the only latency that varies is T_{on}, i.e., the number of cycles required to switch a buffer on. To provide a fair comparison, the same parameters are also used when evaluating the state-of-the-art fine-grained power-gating methodology presented in [6]. For *Power Punch* [3], the T_{on} latency is constant at 8 cycles – as also assumed in [3] – in all experiments in this article. This latency is considerably higher than the T_{on} latency of a single buffer, since it refers to the time required to power up an entire router.

4.2. Results using synthetic traffic

This section discusses the latency, saturation point, and energy profile of the *BlackOut* proposal, considering three different synthetic traffic patterns: uniform random, tornado, and transpose. The *BlackOut* framework is compared against (a) the current state-of-the-art buffer-level (i.e., fine grain) power-gating methodology presented in [6] (henceforth identified as the *Ultra-Fine Grain, UFG,* power-gating scheme); (b) the current state-of-the-art router-level (i.e., coarse grain) power-gating methodology of *Power Punch* [3].

The performance results of the three investigated designs are also juxtaposed to the equivalent results of a baseline NoC, which serves as a performance reference, since it has no switching-off capabilities, and, therefore, no performance penalties are introduced. Similarly, the energy results are normalized to the baseline NoC's energy profile to demonstrate the energy savings reaped by *BlackOut*, *UFG* [6], and *Power Punch* [3].

The aim of this sub-section is two-fold: (1) to demonstrate that router-level (coarse-grain) power-gating proposals lack flexibility, since even in the presence of a single traffic flow from one input-output port pair, the router has to be kept switched on; (2) to demonstrate that *BlackOut* can provide fine-grained control over the switched off resources, thereby extracting better performance under all injection rates up to the network's saturation point.

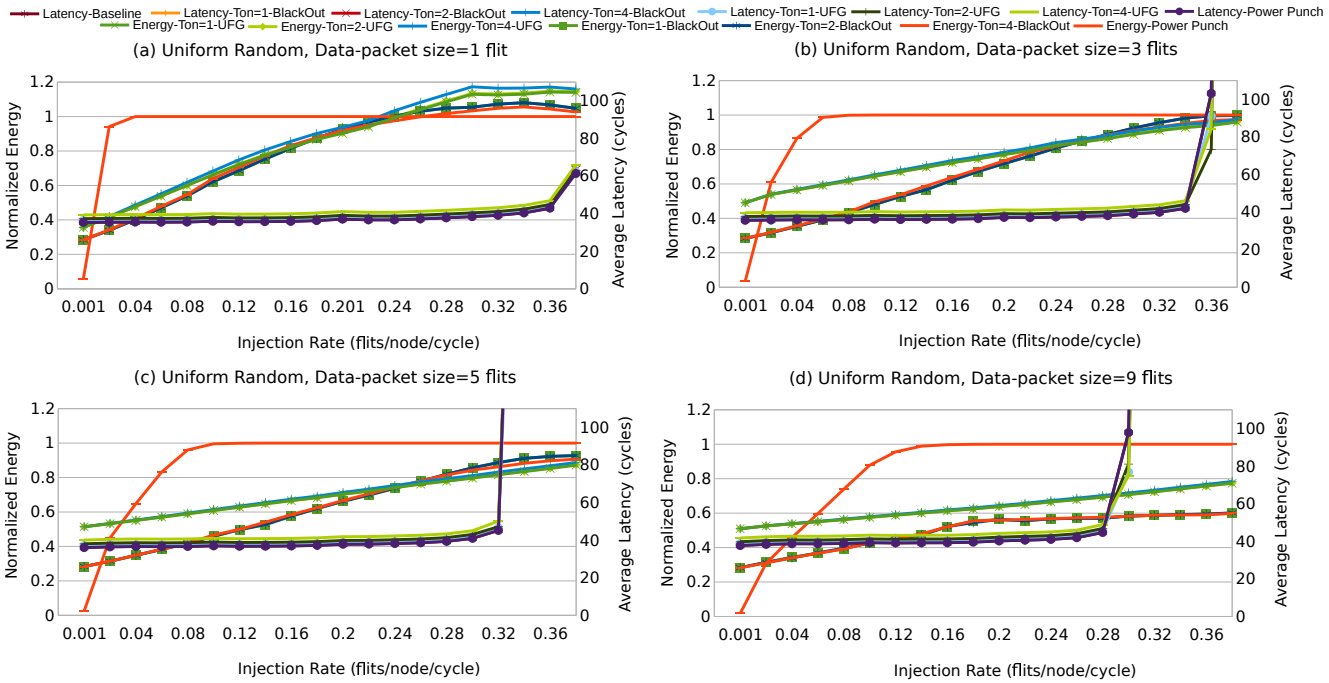


Figure 8: Latency/throughput and energy results using **uniform** random traffic in an 8×8 2D mesh, with 4 different data-packet sizes (1, 3, 5, and 9 flits). Three architectures are compared: *BlackOut*, *UFG* [6], and *Power Punch* [3].

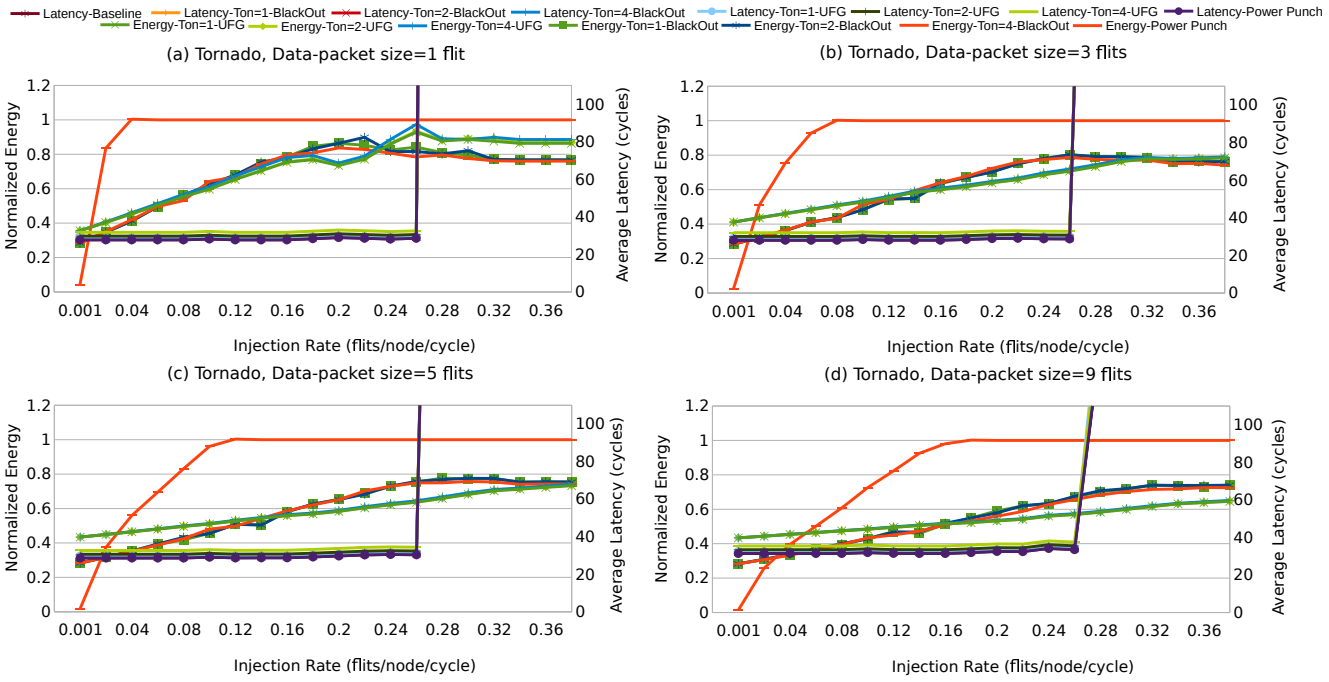


Figure 9: Latency/throughput and energy results using **tornado** traffic in an 8×8 2D mesh, with 4 different data-packet sizes (1, 3, 5, and 9 flits). Three architectures are compared: *BlackOut*, *UFG* [6], and *Power Punch* [3].

Realistic traffic comprises two different packet sizes: (1) *single-flit* packets used for control/coherence messages and cache/memory *requests*, and (2) *multi-flit* packets used for cache/memory *reply* messages (i.e., the packets that contain the actual cacheline requested). Hence, synthetic traffic packets in our simulations are divided into two types: single-flit (*control*) packets and multi-flit (*data*) packets. Control packets are always single-flit, whereas data packets can vary in size, depending on the cacheline size and the NoC's link width. Consequently, the multi-flit *data* packet type is explored further in our simulations. Specifically, we provide results with 1-flit, 3-flit, 5-flit, and 9-flit *data* packet sizes (in separate simulations), in order to account for different cacheline sizes and/or NoCs with different flit widths. In other words, we run simulations with

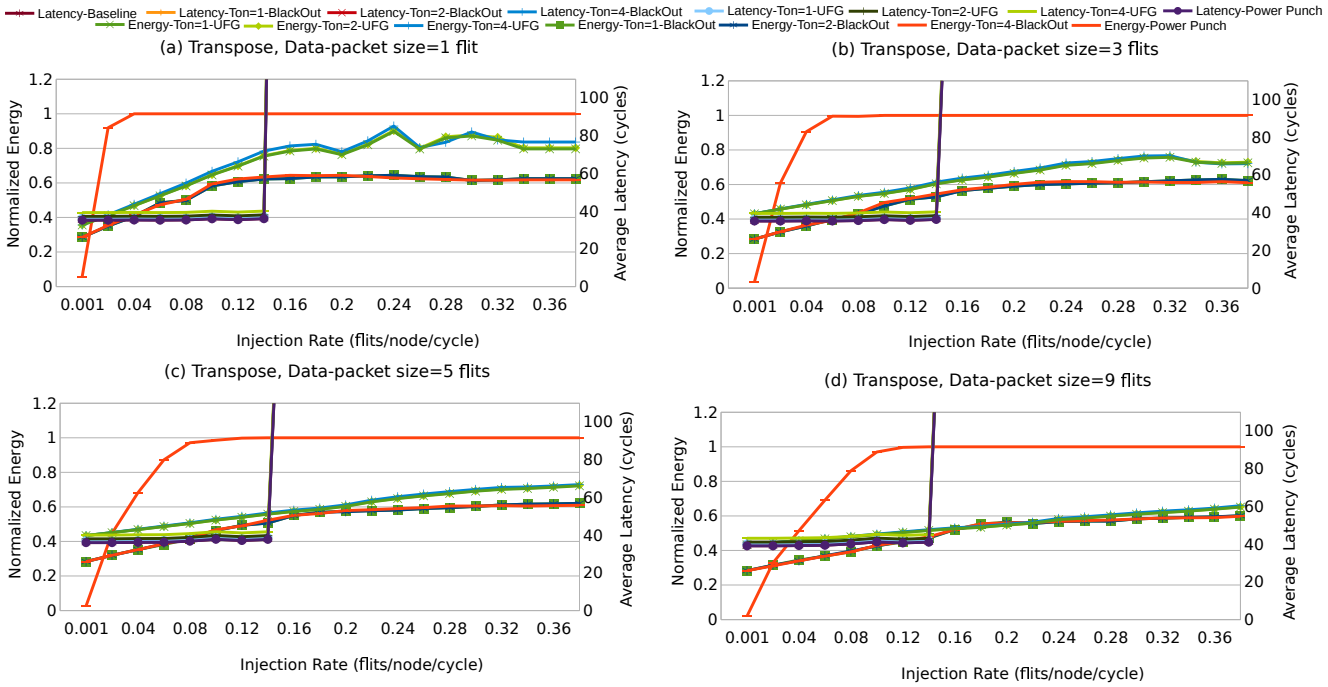


Figure 10: Latency/throughput and energy results using **transpose** traffic in an 8×8 2D mesh, with 4 different data-packet sizes (1, 3, 5, and 9 flits). Three architectures are compared: *BlackOut*, *UFG* [6], and *Power Punch* [3].

(a) *all* single-flit packets, (b) single-flit and 3-flit packets, (c) single-flit and 5-flit packets, and (d) single-flit and 9-flit packets.

The *percentage breakdown* between single-flit and multi-flit packets was chosen to be 66.67% single-flit and 33.33% multi-flit packets. This percentage breakdown aims to mimic the behavior observed in SPLASH-2 benchmark applications [7].

Each simulated scenario considers three distinct VNETs to better resemble the setup required by a realistic cache-coherence protocol in multi-core environments. Two VNETs are always considered to inject single-flit packets, while the third one can inject multi-flit packets (as described above), depending on the selected configuration.

All the figures in the rest of this section share the same format, showing the injection rate in flits/node/cycle on the *x-axis*. The total latency (i.e., network latency plus queuing latency) and the normalized energy consumption are depicted on the left and right *y-axes*, respectively. As previously mentioned, the energy results are *normalized* to the baseline NoC, while the latency results are reported explicitly, with no normalization.

Figure 8 reports the obtained results under uniform random traffic. All three techniques investigated (*BlackOut*, *UFG*, and *Power Punch*) show a saturation point almost identical to the baseline NoC, regardless of the data-packet size, thus highlighting the negligible impact of the power-gating actuator. However, *Power Punch*'s energy profile saturates at 100% (i.e., no energy savings) from very low traffic injection rates, whereas *BlackOut* has an energy profile that grows linearly with the injection rate. This implies that *Power Punch* can obtain the same performance as the baseline NoC, but without any energy savings, except when the injection rate is extremely low, i.e., below 0.04 flits/cycle/node. On the other hand, *BlackOut* provides better performance per consumed energy, since it can accurately maintain the minimum amount of active (switched on) buffers, without incurring any noticeable performance overhead. Note that even at the network's saturation point, *BlackOut* is still slightly below 100% of the normalized energy consumption for multi-flit data packet sizes of 3,5, and 9. This is due to the fact that the depicted results are averaged over the 64 routers, and even if the input port of the traffic generator saturates, the internal NoC router ports are still able to keep some buffers switched off (albeit for a short period of time).

As the size of the multi-flit data packets increases, *Power Punch* behaves in a somewhat counter-intuitively manner. One would expect that larger multi-flit packets would saturate the reaped energy savings at smaller injection rates. However, the opposite trend is observed. This behavior is explained by the fact that the increased size of the multi-flit packets causes a hot-spot-like effect, whereby traffic is bursty and the traversing flits are packed closely together. Consequently, some routers have more opportunities to sleep.

When all traffic consists of single-flit packets, both *BlackOut* and *UFG* consume more energy than the baseline NoC. This is due to elevated numbers of switch-on events. Since switch-on events consume extra energy, an increased number of such events leads to overall energy consumption that is higher than the baseline NoC. When the size of the multi-flit data packets increases to 3 and 5 flits, *BlackOut* outperforms *UFG* by up to 8%. Close to saturation, the two techniques behave similarly, yielding approximately the same amount of energy savings (with *UFG* performing marginally better at saturation). However, the NoC is typically never operated at (and beyond) its saturation point, since performance deteriorates to unusable levels. When the size of the multi-flit data packets increases to

9 flits, *BlackOut* yields 15% more energy savings, on average, than *UFG*.

In terms of performance, *BlackOut* and *UFG* behave similarly, incurring a negligible increase in network latency. Overall, *BlackOut* yields 29% higher energy savings, on average, than *Power Punch*, and 8% than *UFG*, while incurring a near-negligible performance overhead of 2%.

Note that *Power Punch* can save more energy than *BlackOut* (due to the former's ability to completely switch off a router, instead of individual buffers) only at extremely low injection rates, i.e., below 0.01.

The results obtained using tornado synthetic traffic are reported in Figure 9. All three mechanisms – *BlackOut*, *UFG*, and *Power Punch* – achieve the same saturation point as the baseline NoC. Once again, for injection rates higher than 0.06 flits/node/cycle, *Power Punch* provides no energy-saving benefits. On the other hand, *BlackOut* is flexible enough to exploit the traffic characteristics and reap energy savings of around 40%, even at the saturation point. In other words, at the saturation point, *BlackOut* consumes only 60% of the energy consumed by the baseline NoC. This substantial energy saving is attributed to the tornado traffic pattern, which mainly stresses horizontal (East/West) inter-router links in a 2D mesh, thereby allowing *BlackOut* to switch off the buffers in the vertical (North/South) input/output ports. Similar to the uniform random traffic results, *BlackOut* again shows better flexibility than *Power Punch*, irrespective of the data packet size. As a result, *BlackOut* achieves almost the same performance as the baseline NoC, but with massive energy savings (under all injection rates up to the saturation point). Note that *UFG* can sometimes reap even higher – albeit not by much – energy savings than *BlackOut* under tornado traffic. Recall that *BlackOut* always leaves one VC buffer in each input port switched on, to minimize the impact on performance. Instead, *UFG* is able to switch off *all buffers* in the two input ports that are always idle under tornado traffic (North/South ports). In summary, under tornado traffic, *BlackOut* yields 28% higher energy savings, on average, than *Power Punch*, while containing the performance overhead to merely 3%. Despite *UFG's* ability to switch off more buffers than *BlackOut* under tornado traffic, *BlackOut* can still reap the same energy savings as *UFG*, on average.

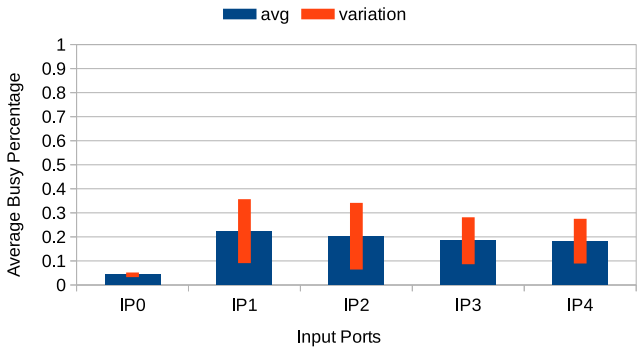
The same trends are observed in Figure 10, which presents the simulation results when using transpose traffic. The three methodologies exhibit the same performance as the baseline NoC, but *BlackOut* can also achieve significant energy savings under all injection rates, up to the saturation point. This is a result of better exploitation of both the traffic shape and the routing algorithm's properties. In particular, when using the XY routing algorithm in conjunction with transpose traffic in a 2D mesh, the stress is concentrated on a relatively small number of NoC links. Thus, *BlackOut* is able to aggressively switch off the large numbers of unused buffers. *UFG* can also take advantage of these traffic characteristics, but not as effectively as *BlackOut*. *BlackOut* can achieve, on average, 10% lower energy consumption than *UFG*, over all data packet sizes. Overall, under transpose traffic, *BlackOut* incurs a minimal 2% performance overhead, while reaping 38% higher energy savings, on average, than *Power Punch*.

4.3. Results using SPLASH-2 benchmark applications

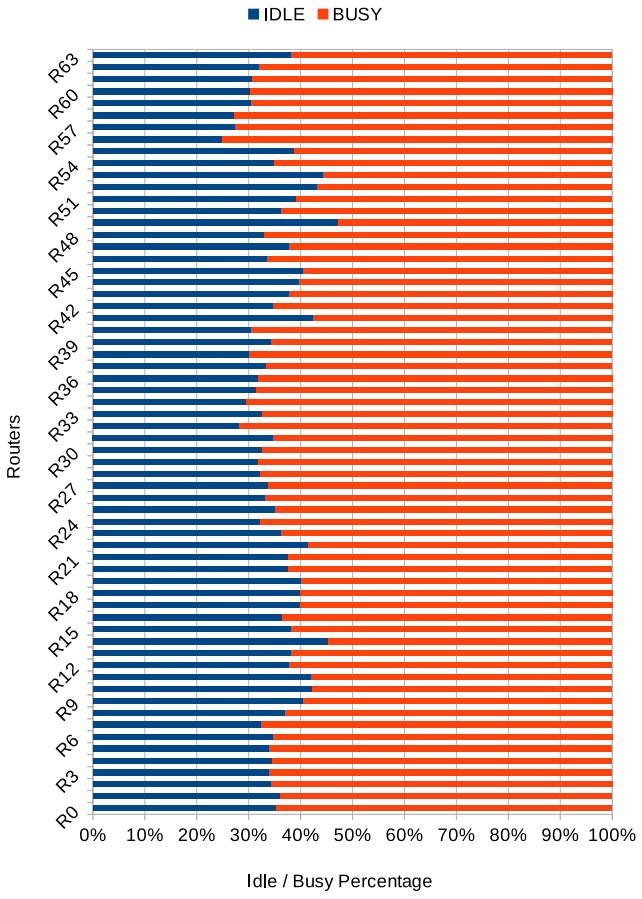
This section evaluates the performance and energy-saving attributes of *BlackOut* when using real application workloads from the SPLASH-2 benchmark suite [7]. We still consider an 8×8 2D mesh multi-core setup. Similarly to the previous sub-section, the *BlackOut* framework is compared against all state-of-the-art power-gating methodologies, i.e., *UFG* [6] and *Power Punch* [3].

The goal of this section is to demonstrate that *BlackOut* can outperform state-of-the-art buffer-level power-gating techniques, due to its sophisticated, yet simple, micro-architectural design features, as described in Section 3. Both *BlackOut* and *UFG* implement the same optimization mechanism discussed in [6], which further reduces the number of switched-on buffers at the local ports of cache-bank nodes. The rationale behind this optimization is that cache nodes are passive entities from the perspective of the NoC; they merely respond to incoming requests. Thus, both *BlackOut* and *UFG* do not need to always maintain a single switched-on buffer in the input port connected to cache-bank nodes. Instead, a buffer is switched on when a request arrives at the node. For the rest of this sub-section, when the above-mentioned optimization is applied to a methodology, the suffix *opt* is appended to the corresponding methodology's name. Before proceeding with the comparison of the three mechanisms, it is worth investigating (a) the percentage of time that the input ports of a NoC router are busy, and (b) the percentage of time that the entire router is completely idle, during the execution of real benchmark applications. This analysis will demonstrate the opportunities for the power-gating of buffers, and it will set the stage for this sub-section's experiments. Figure 11 shows experimental results pertaining to the two above-mentioned investigations. More specifically, Figure 11a shows the percentage of time that the input ports of a specific NoC router are busy (i.e., they cannot be switched off). These results refer to a NoC router situated in the middle of an 8×8 2D mesh, while serving traffic from the *ocean* benchmark, which is used here as a representative SPLASH-2 example application for these particular investigations. Figure 11a indicates an 83% input-port idleness, on average, thereby demonstrating enough potential for independent and fine-grained power gating within each input port. Obviously, a coarse-grained power-gating methodology would be forced to keep the entire router switched on, even if a single input port is active at a given time. Figure 11b shows the percentage of time that the entire router is completely idle. Results for all 64 routers of an 8×8 2D mesh are reported, while serving traffic from the *ocean* benchmark (SPLASH-2). The figure indicates a 33% router idleness, on average.

After establishing the potential for power-gating activity, we now compare the three competing mechanisms using real applications. Figure 12 presents the performance results of *BlackOut*, *UFG*, and *Power Punch*, normalized to the baseline NoC. Three different T_{on} latencies (i.e., the number of cycles required to switch on a buffer) are considered for the *BlackOut* and *UFG*



(a) The percentage of time that the input ports of a specific NoC router are busy. These results refer to a NoC router situated in the middle of an 8×8 2D mesh.



(b) The percentage of time that the entire router is completely idle. Results for all 64 routers of an 8×8 2D mesh are reported.

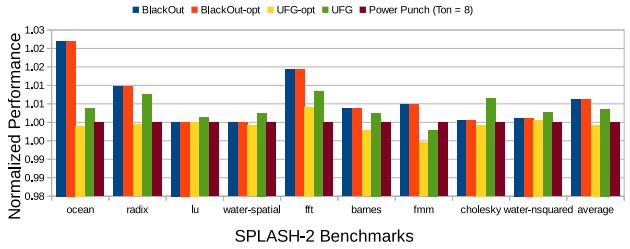
Figure 11: The percentage of time that the input ports of a NoC router are busy, and the percentage of time that the entire router is completely idle. In these experiments, the NoC is serving the traffic generated from the execution of the *ocean* benchmark (SPLASH-2).

methodologies: 1, 2, and 4 cycles. *Power Punch* is only simulated with a T_{on} latency of 8 cycles, as used in the *Power Punch* paper [3]. Any higher T_{on} latencies cannot be masked by the *Power Punch* mechanism, whereas lower T_{on} latencies are not sufficiently long to enable an entire NoC router to wake up. As seen in Figure 12, the incurred performance overhead by the *BlackOut* and *UFG* techniques is limited to within 3%, under all examined benchmarks, regardless of the applied T_{on} latency. Hence, both fine-grained power-gating solutions have minimal impact on system performance. When increasing the T_{on} latency from 1 to 4 cycles, only a negligible increase in the performance overhead is observed, which validates the robustness of the proposed *BlackOut* solution. Furthermore, the optimization technique discussed above, which further reduces the number of switched-on buffers, does not seem to affect the performance of either fine-grained methodology. Specifically, the optimization technique incurs a negligible additional performance penalty, which does not exceed 1%.

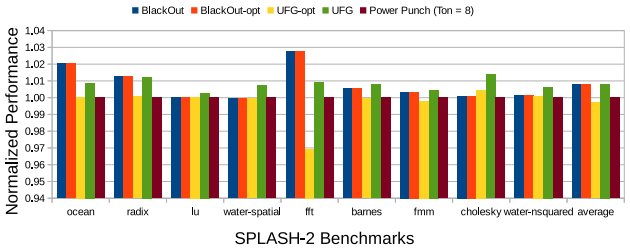
In terms of performance, *Power Punch* behaves as expected, i.e., it does not suffer from any performance degradation, as illustrated in Figure 12. This is due to the fact that Power Punch almost always hides wake-up latencies through the use of so called *slack* (taking advantage of packetization latency within the NIC, and coherence protocol events, in order to mask wake-ups).

Figure 13 presents the energy consumption results, normalized to the baseline NoC, and assuming – for *BlackOut* and *UFG* – T_{on} latencies of 1,2, and 4 clock cycles. Once again, *Power Punch* is only simulated with a T_{on} latency of 8 cycles. Clearly, *BlackOut* substantially outperforms *UFG* in terms of reaped energy savings. In fact, *BlackOut* achieves, on average, 36% lower energy consumption than *UFG*. Figure 13 clearly demonstrates that the energy savings achieved by *BlackOut* are near-identical when the T_{on} latency changes from 1, to 2, to 4 cycles. The same observation applies to *UFG*. It is interesting to note that *BlackOut* yields better energy savings than even the optimized version of *UFG* (i.e., *UFG-opt*). When compared to the regular *BlackOut* scheme, *BlackOut-opt* provides an additional 3% improvement in energy savings.

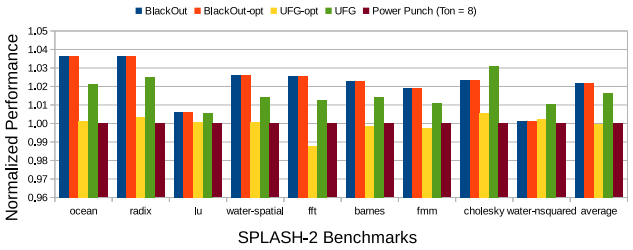
On the other hand, *Power Punch* exhibits very high variability in its energy-saving effectiveness, based on the application's traffic behavior (see Figure 13). For example, the benchmark applications *lu* and *cholesky* generate minimal traffic within the NoC, and said generated traffic is localized (concentrated) within a very small subset of the network. Consequently, *Power Punch* is able to switch off most routers for very long periods of time. While *Power Punch* switches off the *entire* router, *BlackOut* must leave the following components switched on: (a) one VC buffer per input port, (b) the arbitration logic, and (c) the router crossbar. Hence, under such benchmarks (four in total), *Power Punch* can achieve higher energy savings than *BlackOut*. Note, however, that the savings achieved by *BlackOut* under these benchmarks are still very significant. Moreover, if *BlackOut* is augmented with the ability to switch off the crossbar when no traffic is present within a router, then the energy savings will more closely approach those of a coarse-grained power-gating mechanism. In other benchmark applications, like *ocean* and *barnes*, *BlackOut* yields substantially higher energy savings than *Power Punch*. Considering all 9 benchmark applications, the optimized version of *BlackOut* obtains – on average – 3% higher energy savings than *Power Punch*. More importantly, *BlackOut's* behavior is very consistent (with minimal variance) throughout all benchmarks (energy savings ranging from 73% to 75%), while *Power Punch* yields widely vary-



(a) T_{on} = 1 cycle (for *BlackOut* and *UFG*)

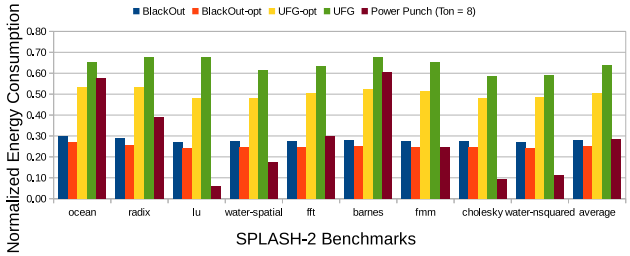


(b) T_{on} = 2 cycles (for *BlackOut* and *UFG*)

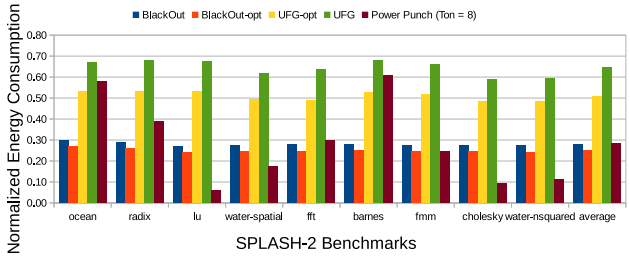


(c) T_{on} = 4 cycles (for *BlackOut* and *UFG*)

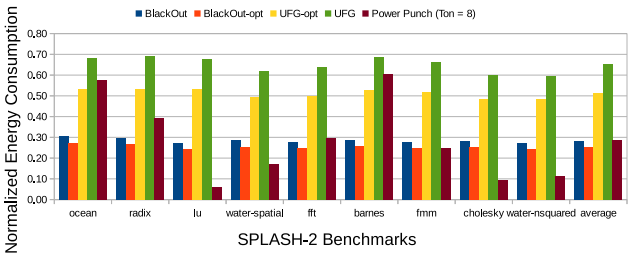
Figure 12: **Performance** results of *BlackOut*, *UFG*, and *Power Punch* when using SPLASH-2 benchmark applications [7]. The results are normalized to the baseline NoC. Three different T_{on} latencies (i.e., the number of cycles required to switch on a buffer) are considered for the *BlackOut* and *UFG* methodologies: 1, 2, and 4 cycles. *Power Punch* is only simulated with a T_{on} latency of 8 cycles. The suffix *opt* denotes the addition of another power-gating optimization technique, as presented in [6].



(a) T_{on} = 1 cycle (for *BlackOut* and *UFG*)



(b) T_{on} = 2 cycles (for *BlackOut* and *UFG*)



(c) T_{on} = 4 cycles (for *BlackOut* and *UFG*)

Figure 13: **Energy consumption** results of *BlackOut*, *UFG*, and *Power Punch* when using SPLASH-2 benchmark applications [7]. The results are normalized to the baseline NoC. Three different T_{on} latencies (i.e., the number of cycles required to switch on a buffer) are considered for the *BlackOut* and *UFG* methodologies: 1, 2, and 4 cycles. *Power Punch* is only simulated with a T_{on} latency of 8 cycles. The suffix *opt* denotes the addition of another power-gating optimization technique, as presented in [6].

ing energy savings, ranging from 39% all the way to 94%.

The SPLASH-2 benchmark applications exhibit *average* injection rates ranging from 5% to 15%. For the applications at the low end of this spectrum, *Power Punch* is very effective. For the applications at the high end of the SPLASH-2 injection-rate spectrum, *BlackOut* becomes much more effective. Obviously, as the traffic intensity increases even more (see synthetic traffic results in previous sub-section), the effectiveness of coarse-grained power-gating approaches quickly diminishes. On the contrary, fine-grained power-gating mechanisms can behave consistently well under all traffic conditions.

4.4. Hardware Implementation Analysis

The *BlackOut* architecture has been fully implemented in RTL SystemVerilog, and fully integrated within a NoC router, which was also implemented in RTL SystemVerilog. The design was synthesized using the Cadence RTL Compiler tool and the FreePDK45 45nm standard-cell library (*https://projects.si2.org/openeda.si2.org/projects/nangatelib*). The router used is a 4-stage baseline NoC router with 5 input/output ports (amenable to a 2D NoC mesh). Each input port has 6 VCs, with each VC buffer having a depth of 4 flits.

The design was synthesized at a frequency of 667 MHz – i.e., the critical path was determined to be at 1.5 ns – and simulated with NC-Verilog (post-synthesis) using a traffic trace extracted from the GEM5 simulator. The traffic trace is from uniform random traffic in an 8×8 NoC, with an injection rate of 0.08 flits/node/cycle (i.e., aligned with the observed NoC traffic injection rates observed under SPLASH-2 workloads). In particular, we considered the traffic trace traversing one of the routers in the middle of the 64-node 2D mesh. The Voltus power tool was used to extract the power trace from the VCD file generated from the simulation, and the average power was subsequently computed.

This setup was used to extract the hardware area, timing, and power overhead of *BlackOut* (with respect to the baseline router). The *BlackOut* architecture was reported to have a critical path of 695 *ps*, i.e., less than half of the router's critical path. More importantly, *BlackOut* operates completely in parallel with the buffer-write stage of the NoC router; thus, it

does not introduce any timing penalty, since *BlackOut's* delay is completely hidden by the much higher clock period of the router. Furthermore, the area and power overheads incurred by *BlackOut* are minimal, at 0.44% and 1.18%, respectively. In terms of absolute numbers, *BlackOut* consumes an area of 300 μm^2, a power of 26 μW, and its critical path (as previously mentioned) is 695 ps.

5. Conclusions

The advent of the multi-/many-core paradigm has elevated the criticality of the NoC. Being an integral part of the system, the NoC consumes non-negligible amounts of power. Consequently, to enable scalability to larger systems, it is imperative to contain the interconnect's power envelope. Out of all constituent components, the router buffers are the major static power consumers within the NoC.

This work has proposed a novel fine-grained power-gating methodology, aptly called *BlackOut*, which can operate at the granularity of individual VC buffers within each input port. The new mechanism entails minimal changes to the router architecture, and it is generally applicable to any pipeline micro-architecture, any topology, and any routing algorithm. The new methodology is fully distributed, and – unlike competing approaches – it only requires minimal information exchange between adjacent router pairs. The *BlackOut* technique relies on a brace of elemental micro-architectural concepts that operate in tandem: *late binding*, and *flow balancing*. Flow balancing enables smooth reactions within the power-gating actuator, which minimize the impact on performance. Late binding re-maps new incoming packets to already-on buffers, thereby avoiding unnecessary wake-ups of "sleeping" buffers.

Extensive evaluations using both synthetic traffic and real application workloads validate the effectiveness of the *BlackOut* methodology. Compared to a baseline NoC, *BlackOut* can achieve energy savings of up to 70%, with a minimal 2% performance overhead. Most importantly, the new mechanism is compared to two state-of-the-art techniques, which represent the leading solutions in router- and buffer-level power-gating granularities. *BlackOut* is demonstrated to significantly outperform both techniques, by 35%, on average, in terms of energy savings.

6. Acknowledgments

This research work has been partially supported by the European Community Seventh Framework Programme (FP7/2007-2013) under agreement no. 612069 (HARPA project http://www.harpa-project.eu/), and by a 3-month HiPEAC Collaboration Grant (https://www.hipeac.net/mobility/collaborations/).

References

[1] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, S. Borkar, A 5-ghz mesh interconnect for a teraflops processor, Micro, IEEE 27 (5) (2007) 51–61. doi:10.1109/MM.2007.4378783.

[2] R. Das, S. Narayanasamy, S. Satpathy, R. Dreslinski, Catnap: Energy proportional multiple network-on-chip, in: Proceedings of the 40th Annual International Symposium on Computer Architecture, 2013, pp. 320–331. doi:10.1145/2485922.2485950.

[3] L. Chen, D. Zhu, M. Pedram, T. Pinkston, Power punch: Towards non-blocking power-gating of noc routers, in: IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), 2015, pp. 378–389. doi:10.1109/HPCA.2015.7056048.

[4] M. Galles, Spider: A high-speed network interconnect, Micro, IEEE 17 (1) (1997) 34–39.

[5] C. Sun, C. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L. Peh, V. Stojanovic, Dsent - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling, in: 6th IEEE/ACM International Symposium on Networks on Chip (NoCS), 2012, pp. 201–210. doi:10.1109/NOCS.2012.31.

[6] H. Matsutani, M. Koibuchi, D. Ikebuchi, K. Usami, H. Nakamura, H. Amano, Performance, area, and power evaluations of ultrafine-grained run-time power-gating routers for cmps, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 30 (4) (2011) 520–533. doi:10.1109/TCAD.2011.2110470.

[7] S. Woo, M. Ohara, E. Torrie, J. Singh, A. Gupta, The splash-2 programs: characterization and methodological considerations, in: 22nd Annual International Symposium on Computer Architecture, 1995. Proceedings., 1995, pp. 24–36.

[8] A. Samih, R. Wang, A. Krishna, C. Maciocco, C. Tai, Y. Solihin, Energy-efficient interconnect via router parking, in: IEEE 19th International Symposium on High Performance Computer Architecture (HPCA2013), 2013, pp. 508–519. doi:10.1109/HPCA.2013.6522345.

[9] D. Zoni, W. Fornaciari, A sensor-less nbti mitigation methodology for noc architectures, in: SOC Conference (SOCC), 2012 IEEE International, 2012, pp. 340–345. doi:10.1109/SOCC.2012.6398329.

[10] D. Zoni, W. Fornaciari, Nbti-aware design of noc buffers, in: Proceedings of the 2013 Interconnection Network Architecture: On-Chip, Multi-Chip, IMA-OCMC '13, ACM, New York, NY, USA, 2013, pp. 25–28. doi:10.1145/2482759.2482766.
URL `http://doi.acm.org/10.1145/2482759.2482766`

[11] D. Zoni, L. Borghese, G. Massari, S. Libutti, W. Fornaciari, Test: Assessing noc policies facing aging and leakage power, in: EUROMICRO DSD/SEAA, Funchal, Madeira, PORTUGAL, Aug 26-28, 2015, pp. 1–8.

[12] L. Chen, T. Pinkston, Nord: Node-router decoupling for effective power-gating of on-chip routers, in: 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2012, pp. 270–281. doi:10.1109/MICRO.2012.33.

[13] L. Chen, L. Zhao, R. Wang, T. Pinkston, Mp3: Minimizing performance penalty for power-gating of clos network-on-chip, in: IEEE 20th International Symposium on High Performance Computer Architecture (HPCA), 2014, pp. 296–307. doi:10.1109/HPCA.2014.6835940.

[14] H. Bokhari, H. Javaid, M. Shafique, J. Henkel, S. Parameswaran, dark-noc: Designing energy-efficient network-on-chip with multi-vt cells for dark silicon, in: 51st ACM/EDAC/IEEE Design Automation Conference (DAC), 2014, pp. 1–6. doi:10.1145/2593069.2593117.

[15] D. Zoni, W. Fornaciari, Sensor-wise methodology to face nbti stress of noc buffers, in: Design, Automation Test in Europe Conference Exhibition (DATE), 2013, 2013, pp. 1038–1043. doi:10.7873/DATE.2013.216.

[16] G. Kim, J. Kim, S. Yoo, Flexibuffer: Reducing leakage power in on-chip network routers, in: 48th ACM/EDAC/IEEE Design Automation Conference (DAC), 2011, pp. 936–941.

[17] M. Casu, M. Yadav, M. Zamboni, Power-gating technique for network-on-chip buffers, Vol. 49, 2013, pp. 1438–1440. doi:10.1049/el.2013.3225.

[18] S. Hassan, S. Yalamanchili, Centralized buffer router: A low latency, low power router for high radix nocs, in: 7th IEEE/ACM International Symposium on Networks on Chip (NoCS), 2013, pp. 1–8. doi:10.1109/NoCS.2013.6558397.

[19] R. Parikh, R. Das, V. Bertacco, Power-aware nocs through routing and topology reconfiguration, in: 51st ACM/EDAC/IEEE Design Automation Conference (DAC), 2014, pp. 1–6.

[20] D. J. Sorin, M. D. Hill, D. A. Wood, A primer on memory consistency and cache coherence, Synthesis Lectures on Computer Architecture 6 (3) (2011) 1–212.

[21] J. Lee, C. Nicopoulos, H. G. Lee, J. Kim, Tornadonoc: A lightweight and scalable on-chip network architecture for the many-core era, ACM Transactions on Architecture and Code Optimization (TACO) 10 (4) (2013) 56.

[22] N. Agarwal, T. Krishna, L.-S. Peh, N. Jha, Garnet: A detailed on-chip network model inside a full-system simulator, in: IEEE International Symposium on Performance Analysis of Systems and Software. ISPASS 2009., 2009, pp. 33–42. doi:10.1109/ISPASS.2009.4919636.

[23] D. Zoni, F. Terraneo, W. Fornaciari, A dvfs cycle accurate simulation framework with asynchronous noc design for power-performance optimizations, Journal of Signal Processing Systems (2015) 1–15doi:10.1007/s11265-015-0989-1.
URL http://dx.doi.org/10.1007/s11265-015-0989-1

[24] D. Zoni, W. Fornaciari, Modeling dvfs and power gating actuators for cycle accurate noc-based simulators, Journal of Emerging Technologies in Computing Systems (2015) 1–15.

[25] N. Binkert, B. Beckmann, G. Black, S. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. Hill, D. Wood, The gem5 simulator, SIGARCH Comput. Archit. News 39 (2) (2011) 1–7. doi:10.1145/2024716.2024718.
URL http://doi.acm.org/10.1145/2024716.2024718