ShuttleNoC: Power-Adaptable Communication Infrastructure for Many-Core Processors

Hang Lu, Yisong Chang, Guihai Yan, *Member, IEEE*, Ning Lin, Xin Wei, and Xiaowei Li, *Senior Member, IEEE*

***Abstract*—Networks-on-chip (NoCs), as the communication infrastructure in many-core processors, has demonstrated remarkable power consumption along with the technology scaling. However, due to the temporal and spatial heterogeneity of the on-chip traffic, one critical problem is that the NoC power consumption cannot effectively adapt to the variation of its traffic intensity, also known as localized power adaptation, hence yielding a suboptimal power efficiency. Prior approaches either resort to the over-provisioned NoC design or coarse-grained bandwidth scaling to partially alleviate excessive power consumption brought by the traffic temporal or spatial heterogeneity. While in this paper, we propose a novel NoC architecture called Shuttle NoC (ShuttleNoC) to address this challenge. It leverages the link reconfiguration to enable flexible packet traversing between multiple subnetworks, and specialized punch lines to accelerate latency sensitive traffic. With the support of the dedicated power adaptation mechanisms, it is shown in the evaluation that the proposed ShuttleNoC architecture could effectively tackle the power and performance tradeoff and significantly boost the power efficiency compared with the state-of-the-art baselines.**

***Index Terms*—Many-core processors, networks-on-chip (NoCs), power management, shuttle networks-on-chip (ShuttleNoC).**

I. Introduction

ALONG with the rapid growth of chip integration, power consumption has become a first-order design constraint in modern many-core processors. Ever-growing power consumption not only leads to an increased energy and packaging costs, but also results in high die temperatures that may, in the worst case, jeopardize the chip performance and reliability.

The power consumption of a many-core processor can be roughly broken down into the computation power by cores and the communication power by networks-on-chip (NoCs). Recent studies show that the power consumption of NoC could reach as high as 80 Watts, a large slice of the total chip power, under 16-nm technology node for a mesh connected multicore [1]. The same trend also emerges at other commercial designs, i.e., Sun's "Niagara" processor, the interconnect takes nearly 17% of the total power consumption [2]; this percentage reaches up to 28% in the "Intel80" processor [3]. In future many-core era, the NoC power consumption is expected to increase more rapidly [4]–[6], because most computation resources are restricted inactive due to the dark silicon problem. However, the communication infrastructure must be kept alive to serve the memory accesses, so the proportion of its power consumption is enlarged that is even on par with its computation counterpart [7]–[10].

Compared with the computation power which could be regulated with many core-level techniques such as dynamic voltage and frequency scaling (DVFS), communication power is more sophisticated to be managed, and without power efficient communication infrastructures, we have little chance to hit the power efficiency frontier at the chip level. The major bane of optimizing communication power stems from achieving *localized power adaptation*, which means each node (router + link) should deliver proportional power quota in accordance with its local traffic intensities during workload executes [5], [11]–[14]. This requirement, though intuitively simple, is hard to accomplish, given the sporadic traffic distributions in not only temporal, but also spatial dimension. Worse still, the on-chip traffic intensity may vary at the nanoseconds magnitude [15] and across unexpected locations of the chip [16]–[18]. Chip designers cannot easily customize the NoC power consumption to adapt such variations.

To address this challenge, recent studies aim at reducing the power of the major components like on-chip routers and links by deploying DVFS or power gating, clocking gating in certain voltage and frequency islands (VFIs) [19]–[21]. Although deploying these power control schemes in the network would substantially reduce the power consumption of the entire NoC, careless regulating the power states of these components may also severely malign network or even system performance. In specific, applying DVFS in the VFIs relies on the complex synchronization at the VFI boundaries. In large-scale many-core processors running multiprogram workloads, communication traffic will inevitably traverse considerable amounts of VFIs to the target memory controller or the shared last level cache, which will introduce remarkable synchronization latency [19] limiting the headroom of network performance.

On the other hand, some approaches [6], [12], [22]–[24] propose to use power gating to the on-chip routers to avoid the

Manuscript received March 1, 2018; revised May 21, 2018; accepted June 29, 2018. Date of publication July 12, 2018; date of current version July 17, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61532017, Grant 61432017, Grant 61522406, Grant 61572470, Grant 61521092, Grant 61602442, and Grant 61702485, and in part by the Youth Innovation Promotion Association, CAS under Grant Y404441000. This paper was recommended by Associate Editor Y. Wang. *(Corresponding authors: Guihai Yan; Xiaowei Li.)*

The authors are with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, University of Chinese Academy of Sciences, Chinese Academy of Sciences, Beijing 100190, China (e-mail: yan@ict.ac.cn; lxw@ict.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2018.2855165

frequency switches between VFIs, as a way to circumvent the overhead of frequent cross-domain synchronizations. However, hasty activating or shutting down routers on the data path will turn the previous regular network topology into irregular shapes, also known as the *connectivity* problem. In-transit packet must detour the sleeping routers that not only increases the distance to the destination, but also increases the complexity of the underlying routing mechanism. It requires that the routing must be reconfigurable since the candidate routers to be deactivated are highly unpredictable that could exist at any location of the chip. Moreover, activating a sleepy router is not instantaneous. Traffic must wait at the upstream router for the downstream router to be completely activated [23], and will be inflicted with enormous contention delay. The same problem also happens at the network injecting and receiving phase, as the local router is disconnected from the local memory subsystem due to power gating. This scenario is more frequently encountered in traditional NoC designs, because no alternative path could be provided to reach the target memory system, which also means the deployment of power gating becomes more difficult in these architectures.

Since effective power adaptation for the NoC is critical for achieving an optimal system-level power efficiency, in this paper, we propose a novel NoC design, termed as Shuttle NoC (ShuttleNoC), and the associate power adaptation mechanisms to address the aforementioned challenges in tandem. First, apart from traditional "one-case-for-all" or multiple NoC (MultiNoC) design [5], [25]–[28], ShuttleNoC is rearchitected by organizing several implicit subnetworks with equal bandwidth. It is more amenable to power gating without compromising the *connectivity* of the network. Second, ShuttleNoC is able to timely adapt application runtime bandwidth demands and operate itself accordingly by taking both temporal and spatial traffic heterogeneity into account. By monitoring traffic intensity at each node, subrouter and its associate links could be powered on or off to implement the localized power adaptation without losing connectivity or affecting other nodes in the vicinity. Packets in a subnet are allowed to *shuttle* into the downstream active subnets via the specialized link reconfiguration module (LRM), rather than waking up the gated downstream subrouter in the same subnet to proceed. Therefore, a majority of subrouters do not need to stay alive to serve the scarce traffic. As another major feature, latency-sensitive traffic has the opportunity to be accelerated by reserving the dedicated punch lines bridged between neighbors to avoid the worst-case performance degradation.

ShuttleNoC could attain optimal power and performance efficiency, but at the same time, stay free of the issues faced by the traditional NoC designs. Generally speaking, this paper makes the following contributions.

1) We propose ShuttleNoC architecture to fulfill localized power adaptation. We leverage our insights from the weaknesses of the existing heterogeneity-agnostic NoC designs. By leveraging link flexibility, it avoids the unnecessary activation of the subrouters, so the temporal/spatial heterogeneity of the on-chip traffic could be better accommodated.

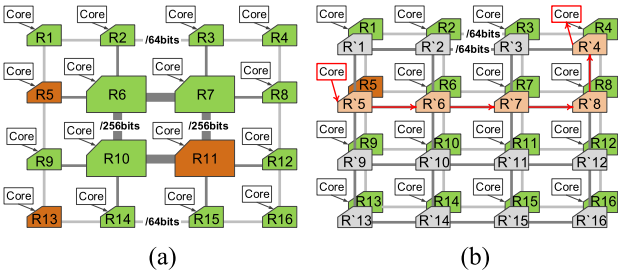


Fig. 1. NoC architectures supporting only spatial or temporal heterogeneity in recently proposed literatures.

2) On top of the ShuttleNoC concept, we present two canonical microarchitecures: 1) ShuttleNoC-Sparse (ShuttleNoC-S) and 2) ShuttleNoC-Dense (ShuttleNoC-D) for different application scenarios. ShuttleNoC-S and -D have different data path complexity and the internal scheduling mechanisms. They each have pros and cons, and could be deployed according to the latency or bandwidth sensitivity of the running workload.
3) We propose a dedicated power adaptation mechanism associated with the ShuttleNoC architecture. As the power subsystem, we design the power-gating and wakeup policies integrated in the router microarchitecture. It equips ShuttleNoC to react quickly to the different network conditions like bursty traffic or traffic already experiencing large delays.

II. Background and Motivation

Heterogeneous communication infrastructure in many-cores is a widely studied topic in recent literatures. Some techniques [4], [8], [29], [30] propose to use several different subnetworks with different voltage and frequency settings to serve bandwidth and latency-sensitive applications. Although they achieve obvious performance improvement, fixed bandwidth at each node is hard to capture traffic heterogeneity in a global manner. In order to maintain network connectivity, Chen and Pinkston [23] proposed a decoupled core-router injection technique. Instead of waiting for the drowsy router to be fully activated, it allows the processing element to inject poised packets into a cyclic bypass path. Thus, blocking delay is able to be, to some extent, alleviated but at the same time, the design complexity of the underlying routing is also augmented which is proportional to the scale of the many-core processors.

Some other prior studies [15] devote to design a heterogeneous NoC based on the traffic "spatial" distribution. For example, in Fig. 1(a), big routers are relatively designed for boosting network performance by providing a higher bandwidth (256 bits), in comparison with small power efficient routers (64 bits). This design philosophy assumes that the routers in the central area of the mesh topology will handle much heavier traffic than those at the boundaries, so a larger power consumption of big routers is essentially expected to obtain a proportional performance enhancement. Whereas, on-chip traffic distribution is never fixed and hotspots may migrate anywhere in NoC, especially when it handles multiprogram

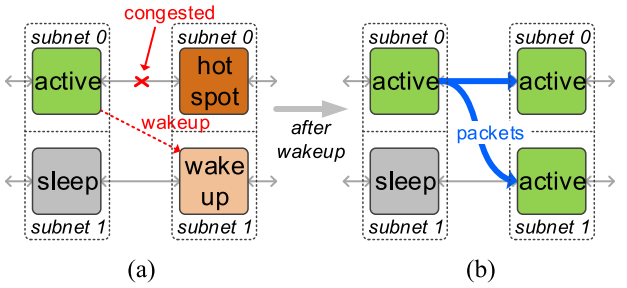


Fig. 2. Motivation example. Power adaptation based on local traffic intensity. In this paper, we term this packet steering as "shuttle." (a) Waking up a higher-level router. (b) Packet shuttling into the newly activated router.

workloads. Small routers like $R5$ and $R13$ may also encounter intense traffic while big routers are almost idle. Under these circumstances, power efficiency will suffer.

At the other end of the spectrum, some solutions aim to design a configurable NoC to capture the traffic "temporal" heterogeneity. MultiNoC [5], as a representative, evenly breaks down the original single NoC into multiple subnetworks (subnets). By deploying power gating, an entire subnet could be powered on or shutdown according to the temporal traffic variations, as shown in Fig. 1(b). However, such temporal-oriented approach ignores the spatial traffic distribution, and is not a comprehensive solution either. For example, in Fig. 1(b), there are two subnets with each 64 bits wide. If a traffic flow intends to traverse from node 5 to node 4 under dimensional order routing, i.e., XY routing, and subrouter $R5$ is already a hotspot, the only solution is to wake up all subrouters along the path in subnet 2 ($R'5$, $R'6$, $R'7$, $R'8$, $R'4$) to serve this traffic flow, even though subrouters $R6$, $R7$, $R8$, $R4$ in subnet 1 are sufficient to handle this traffic flow. Hence, this power management approach degrades the power efficiency of these nodes.

The temporal/spatial heterogeneity yields different network demands and power efficiency consequences. Ideally, we would expect each node's bandwidth to be in line with its local traffic intensity. Power management should be capable to adapt to both temporal and spatial heterogeneity of the on-chip traffic, rather than either side of them. For example, Fig. 2(a) shows a scenario that packets are blocked by a "hotspot" subrouter. If it could "Wakeup" the "higher level" subnet at the neighboring node, and steer the congested packets into the newly activated subrouter, the congestion condition would be effectively alleviated, just as Fig. 2(b) shows. By contrast, if an active subrouter is redundant due to the light traffic condition, we can offload the traffic back into the "lower level" subnet and control the offloaded subrouter to "Sleep" state to save power. We term such packet steering between different subnetworks as "packet shuttle."

The above description motivates a new network design that leverages the packet shuttling to elongate router sleeping cycles and evade the blocking delay on the critical path by keeping the network connectivity. In the next section, we will specify the details of the ShuttleNoC concept, the microarchitectures and the associate power adaptation mechanisms enforced.

III. Shuttle Networks-on-Chip Architecture

A. General Concept

In order to achieve the aforementioned packet shuttling, we need some modifications in terms of the router and link microarchitecture upon traditional NoCs. The proposed ShuttleNoC design is shown in Fig. 3, without loss of generality, we start describing its microarchitecture using a 4×4 mesh connected NoC with two subnets. The figure shows two instances of ShuttleNoC paradigm. Fig. 3(a) shows ShuttleNoC-S, while Fig. 3(b) shows ShuttleNoC-D. The packet shuttling is implemented through two hierarchies.

1) At the chip level, apart from temporal-oriented approach, a particular fraction called LRM is added between neighboring nodes, which makes previously separated subnets related to each other. The link of a subnet is reachable to other subnets after reconfigured in LRM.
2) For an individual subrouter, we need additional control paths connected to the LRM to transmit "shuttle requests," so LRM could reconfigure the links accordingly and steer the flits to the desired subnet. But note that the implementation of LRM differs for ShuttleNoC-S and ShuttleNoC-D, and that is also the major distinction of the two instances, which will be specified later.

In specific, ShuttleNoC resolves two power efficiency limitations associated with previous heterogeneity-agnostic approaches. First, it eliminates the unnecessary activation of the subrouters, hence avoiding the over-provisioned power consumption at the less congested nodes. This benefit, unique in ShuttleNoC, stems from the flexible link connectivity provided by the *sparse* link reconfiguration unit (SRM). Taking the same packet forwarding example in Section II, only $R'5$ is necessary to be activated in ShuttleNoC-S as Fig. 3(a). Packets could shuttle from $R'5$ to $R6$, which is a light-loaded subrouter ($R'6$ is still sleeping), and proceed to the destination in subnet 1. Thus, we have 4 less router activations. For other passing-by packets, i.e., $R1 \rightarrow R9$, $R'5$ could also be used for shuttling, leaving $R'1$ and $R'9$ at sleeping state. Hence, the overall power consumption could be reduced significantly. Second, apart from spatial-oriented approach, the bandwidth of a node is never fixed but could be dynamically coordinated, so the traffic spatial variations could be well adapted to further improve power efficiency.

ShuttleNoC-S divides multiple subnetworks at the node level, that is, $R1$ and $R'1$ belongs to subnet **1** and **2**, respectively, while for the ShuttleNoC-D architecture, the difference is that the subnetworks are divided based on the *location* of the node. In particular, $R1$ and $R'1$ both belong to subnet **1**, but $R2$ and $R'2$, however, belong to subnet **2**. Packets are endorsed to shuttle between subnets ($R1 \rightarrow R'5$), or in other cases between routers within the same subnet ($R1 \rightarrow R'3$). ShuttleNoC-D relies on more complex *dense* reconfiguration module (DRM) residing on the links to achieve more aggressive power adaptation. The major difference compared to ShuttleNoC-S is that it allows "shortcuts" to bypass intermediate node(s), to further minimize router pipeline delay for the latency-sensitive traffic. The shortcuts are directly connected via two adjacent

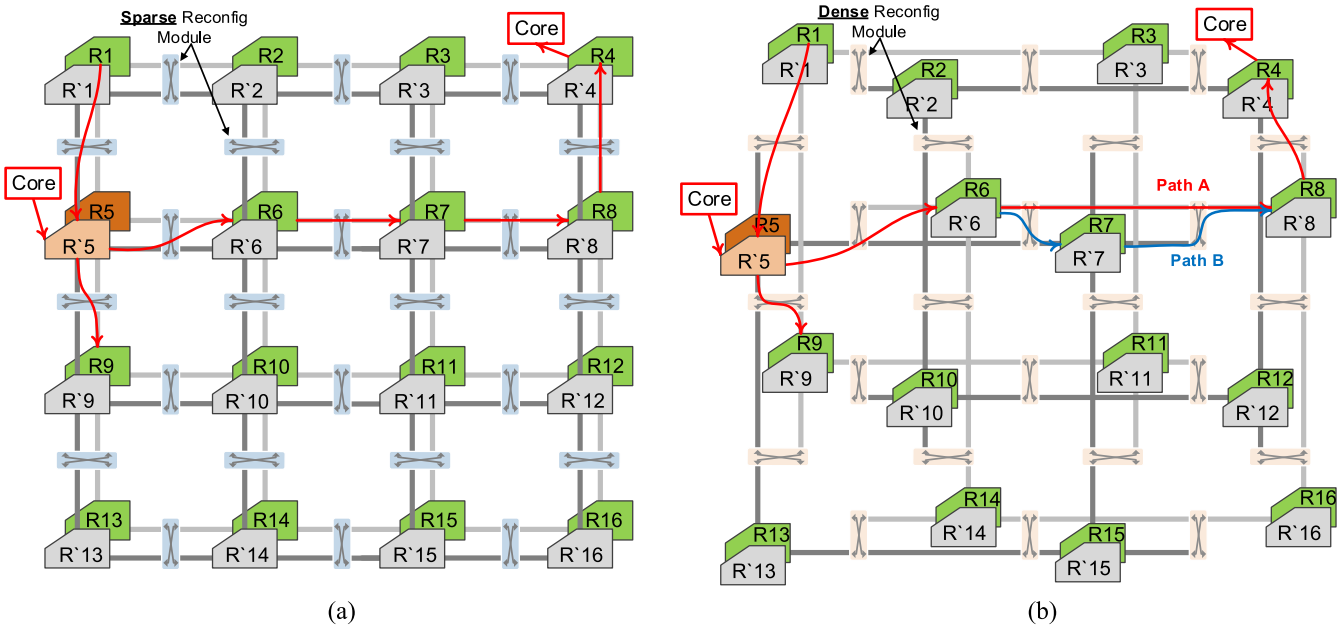


Fig. 3. ShuttleNoC architecture. We show two instances in ShuttleNoC design paradigm. Both constitute of two subnetworks, and instantiate multiple LRMs between the consecutive nodes. The two designs differ in LRM and subnetwork division: ShuttleNoC-S differentiates subnets within each node, while ShuttleNoC-D differentiates each subnet at different nodes.

DRMs, bridged as the express way according to the requests imposed by the upstream subrouters. Taking the same packet as an example, in ShuttleNoC-D, $R\backslash 5$ is still the only activated router to alleviate congestion at $R5$, but after packets are schedule into $R6$, it could bypass $R7$ and reach $R8$ (Path A), so the pipeline delays in node 7 is entirely evaded for this packet. As an alternative, it could also select Path B by steering itself into $R7$ and then back to $R8$, if the virtual channel resources in $R8$ are not available right now or other competitive packet has won the arbitration of using Path A at that time spot.

As an upgraded version of ShuttleNoC, it provides more flexible data path versatilities to address the low power consumption and the optimal packet latency in synergy. However, it is also imperative that more complex dense reconfiguration module must be enrolled to construct the requested data paths, as well as relatively more complex arbitration mechanisms handling concurrent shortcut requests. In the next sections, we will elaborate the microarchitectures of LRMs and the router implementations with the associate packet steering mechanisms, and how they collaborate to support the localized power adaptation.

B. Microarchitectures

1) Link Reconfigure Module: As mentioned before, LRM serving as the key component to achieve packet shuttling, has two different instances for ShuttleNoC-S and ShuttleNoC-D. The detailed implementation of SRM is shown in Fig. 4. In order to implement packet shuttling, we need additional control and data paths between the neighboring nodes. Therefore, SRM is comprised of two strongly coupled yet cost-effective modules: 1) packet steering and 2) multiplexer array each responsible for receiving and arbitrating the proposed requests and constructing the data paths accordingly.

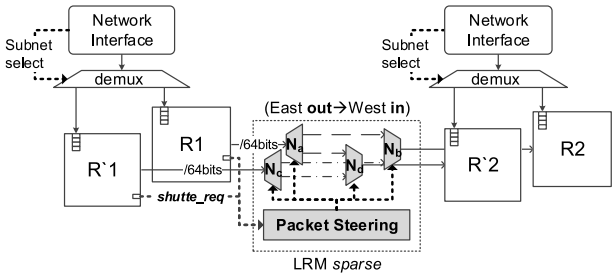


Fig. 4. ShuttleNoC-S data path configuration, (re)constructed by SRM.

For example, at the east output of $R1$ in the figure, subrouter could issue a shuttle request (shuttle_req) for the destination subnet, i.e., west input of $R\backslash 2$. The data path is reconfigured by controlling the multiplexer array in the SRM. According to different enabler combinations, we can get different subnet connections. In Fig. 4, supposing a packet intends to shuttle from $R1$ to $R\backslash 2$, LRM configures the route as $R1 \rightarrow N_d \rightarrow N_d \rightarrow R\backslash 2$. Obviously, although we adopt SRMs on links, it only introduces a negligible power and area overhead in the architecture.

As for ShuttleNoC-D architecture, the data path is augmented with bi-directional shortcuts, as a way to accelerate latency-sensitive packets to bypass the intermediate nodes along its path to the destination. As the major distinction compared with ShuttleNoC-S, we term the express way as *punch line* in this paper, as shown in Fig. 5. The punch line connects two adjacent DRMs with the same bandwidth as normal data path, bridging physically segregated nodes at all four directions. The main benefit brought by such punch line connection is that the selected packet that has already suffered from long contention delay is provided the opportunity

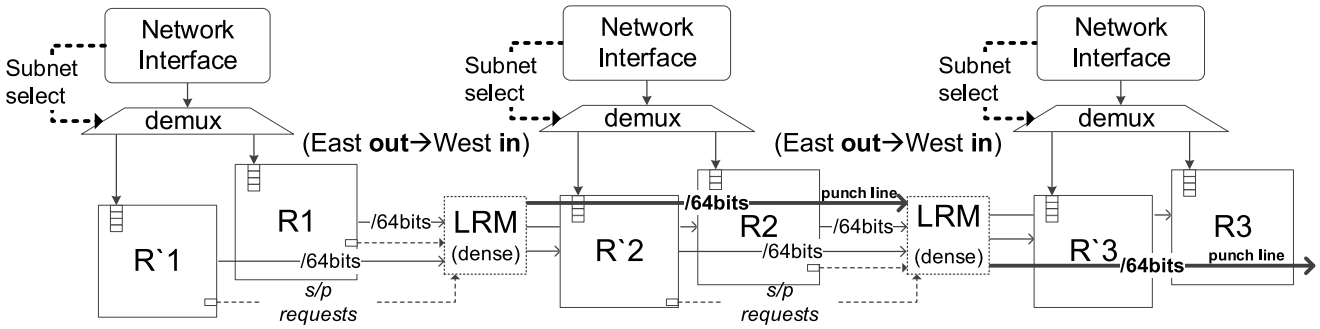


Fig. 5. ShuttleNoC-D data path configuration, governed by the DRM. It enables the latency sensitive packets to surf through the express way (bold links in the figure) and bypass the intermediate nodes. We term these shortcut links as *punch line* in this paper.

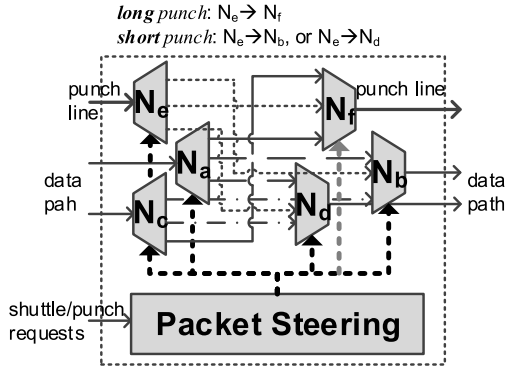


Fig. 6. Details of densely connected LRM (DRM) in ShuttleNoC-D. We have two more MUXs for punching.

to overtaking other packets, which is conductive to minimizing the worst-case latency introduced by the traffic hotspots. Taking the similar example in Figs. 4 and 5 shows three nodes from the East output of $R1$ to the West input of $R3$ and the punch line allows the packet leaping $R2$ and reaching $R3$ in just one link traversal cycle, evading the contention delays in $R2$. We define the punch bypassing one node as *short punch*. Besides, as can be imagined, two adjacent punch lines are separated by one DRM, so it is possible that the packet could reserve the two punch lines simultaneously and prolong the leaping distance. Packet spawned from the East output of $R1$ in Fig. 5 could traverse $R2$ and $R3$ via the two bolded punch lines and approach farther downstream node. We define such operation as *long punch* in this paper.

The punch feature is provided by more fine-grained scheduling in DRM. Fig. 6 shows the circuit design in DRM. Compared with SRM, it introduces two additional multiplexer N_e (whose *input* data path is from *upstream* punch line) and N_f (whose *output* data path is to *downstream* punch line), to select between the normal shuttle path and local punch line. If the traffic intends to reach $R2$ from $R1$, the multiplexers are constructed in the same way as in Fig. 4. However, if the packet has reserved the local punch line to attain $R3$, the DRM configures the route as $R1 \rightarrow N_a \rightarrow N_f \rightarrow$ DRM, and the packet finally reaches $R3$, bypassing intermediate node 2.

2) Router Microarchitectures: Having introduced the infrastructure on the links, we then specify the router microarchitecture by first elaborating the pipeline stages used to support the shuttle or punch operation. ShuttleNoC has typical pipelines as the conventional NoC routers. Meanwhile, it has unique operations that are shadowed with the normal routing pipelines as shown in Fig. 7. The detailed shadowed stages include the following.

1) Punch line arbitration (PA) shadowed with the routing computation (RC), it determines the winner packet that could use the local punch line. The reason it lodges itself with RC is that RC is able to provide the destination of the current packet as well as the distance to its desired destination. As can be imagined, only the distance larger than two nodes is meaningful for this packet to use the local punch line. Otherwise, it means the packet will reach the desired destination right at the next hop, and using punch line will make it falsely miss the destination.
2) Remote VC allocation (RA) shadowed with the virtual channel allocation (VA), the winner packet could use the punch line, so apart from normal VA in the conventional routing pipeline, it is also entitled to reserve remote virtual channels that are at least two nodes away from its current node, to support punch operation.
3) Generate shuttle requests (GS) shadowed with the switch allocation (SA), both instances of ShuttleNoC propagate shuttle/punch request to the local/remote LRM. LRM is responsible to arbitrate amongst multiple concurrent requests at this stage.
4) Punch line reservation (PR) shadowed with switch traversal (ST), this stage reserves the punch line before the final link traversal, informing LRM to configure the path in advance to serve the packet at the next cycle.

The pipeline stages are not exactly identical for ShuttleNoC-S and ShuttleNoC-D. In specific, ShuttleNoC-S only has the RA and GS stage, because ShuttleNoC-S does not architect the punch line in its SRM so it does not need to arbitrate or reserve punch lines. Due to the packet shuttling feature, a local subrouter needs to probe the available virtual channel resources that may belong to a different subnet, which is the major difference compared to the conventional VA stage. Once one of the downstream subrouters is active, its VC occupancies are hence acknowledged by its neighborhood, as shown in Fig. 7. Router needs to record the already allocated VC, together with its subnetwork ID. In SA/GS stage, the operation involves sending this runtime information to the LRM's

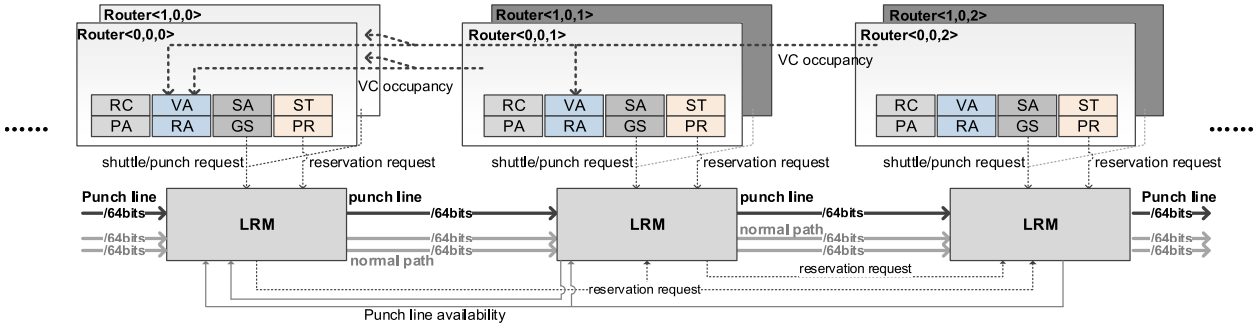


Fig. 7. ShuttleNoC router pipeline stages. All stages are shadowed with the conventional NoC router pipeline. ShuttleNoC-S contains RA and GS, while ShuttleNoCD has all four stages.

"packet steering" module for further arbitration and data path reconstruction. Whereas, for the ShuttleNoC-D, the pipeline stages are relatively more complex than ShuttleNoC-S, involving all four stages. Packets could not only reserve the normal VCs of the next hop if there are credits, but also seize the punch line for acceleration. Therefore, pipeline stages are modified from the very beginning, but note that ShuttleNoC-D router pipelines are coupled with each conventional stage, so it will not introduce additional pipeline latencies.

a) Punch packet selection in the PA stage: Intuitively, multiple packets would impose concurrent reservations to the same punch line of the local node for acceleration, and that will lead to the punch line competition and even traffic congestion if they are not regulated properly. In order to efficiently schedule packets and mitigate the latency introduced by such competitions, DRM maintains an arbitration mechanism in the PA pipeline stage to determine the *long* or *short* punch based on the accumulated in-transit packet latency, with the following metric:

$$\begin{aligned} \alpha\text{Avg} \le \text{Lat}_i \le \beta\text{Avg} &\overset{\text{then}}{\Longrightarrow} \text{short punch} \\ \text{Lat}_i > \beta\text{Avg} &\overset{\text{then}}{\Longrightarrow} \text{long punch} \\ \text{Lat}_i < \alpha\text{Avg} &\overset{\text{then}}{\Longrightarrow} \text{shuttle} \end{aligned}$$

wherein Lat_i is the latency packet i has experienced, represented in cycles. This latency information is embedded within the packet's header flit, and is interpreted for evaluation at the very beginning of the RC/PA stage, while Avg is the sliding average latency value of the historical packets passing through this router. Coefficient α and β are both empirical values that quantize the relationship between Lat_i and Avg and further determine if the packet could use the local punch line or not. We use this straightforward metric because, on one side, different locations have different congestion status and obviously each packet will experience different contention delay, so one packet may be accepted to use the punch line but in other routers only shuttle path will be allocated to it. We want the arbitration is based on the local traffics rather than global information. On the other side, local decision making is relatively easy to implement eliminating congestion propagation paths commonly seen in previous congestion control techniques [16], [17], [31], [32], so it reduces the implementation overhead of the chip. To determine an optimal setting of the two coefficients, we use a cycle-accurate NoC simulator running workload traces to offline train the two coefficients. We also carry out a design space exploration to see the impact of their scaling in Section IV.

b) Punch/shuttle request arbitration in the GS stage: ShuttleNoC is composed of several parallel-organized subnetworks/subrouters at each node. Sometimes, multiple subrouters will impose concurrent requests, i.e., multiple packets intend to shuttle into the same downstream subrouter in the GS stage. Or, they might impose the same punch line requests for acceleration at the PR stage. Therefore, we need two arbitration mechanisms to determine the target packet that could use these shared resources. This time of arbitration, implemented in LRM instead of in each subrouter, is also required to avoid sophisticated microarchitectural design, so we use *round Robin* algorithm, to poll each subrouter if it has posed a request that will be received by the packet steering module in the LRM. For punch line competition, we still employ round robin to allocate punch line accessibility for each subrouter, so each subrouter can fairly use the express way for acceleration if it spawns a punch packet.

c) Detailed pipeline stage transition in the router microarchitecture: As demonstrated in Fig. 8, after PA stage has determined which packet is endowed with the privilege to use the local punch line. Its VC ID is registered as "PL-VCID." While in the RA stage, punch-line packet could reserve remote VCs if available, normal packet in local VCs could, however, only reserve the available VCs in the downstream subrouters. After RA stage, the pipeline enters SA stage in which some special tricks are necessary. SA outputs the winner VC (recorded as "SW-VCID") that could prepare the ST, so the winner might be either punch line VC or normal VC. We employ five comparators at each input VC port and compare if the SW-VCID equals to the PL-VCID in the SA/GS stage. For example, in Fig. 8, we see that if VC1 (marked in blue in input port 4) is the previously recorded PL-VCID in the PA stage and wins in the SA, the comparator initiates the control signal to the corresponding multiplexers selecting between "punch" or "shuttle" request. The result will subsequently be notified to the P_Steering module in the LRM. Combining the type of punch operation (*long* or *short* punch), P_Steering configures the multiplexer array to establish the desired data path. In the next cycle, packet will traverse the link to the target downstream node, which might just be its neighbor or three nodes away after surfing the punch line. We allocate one latch

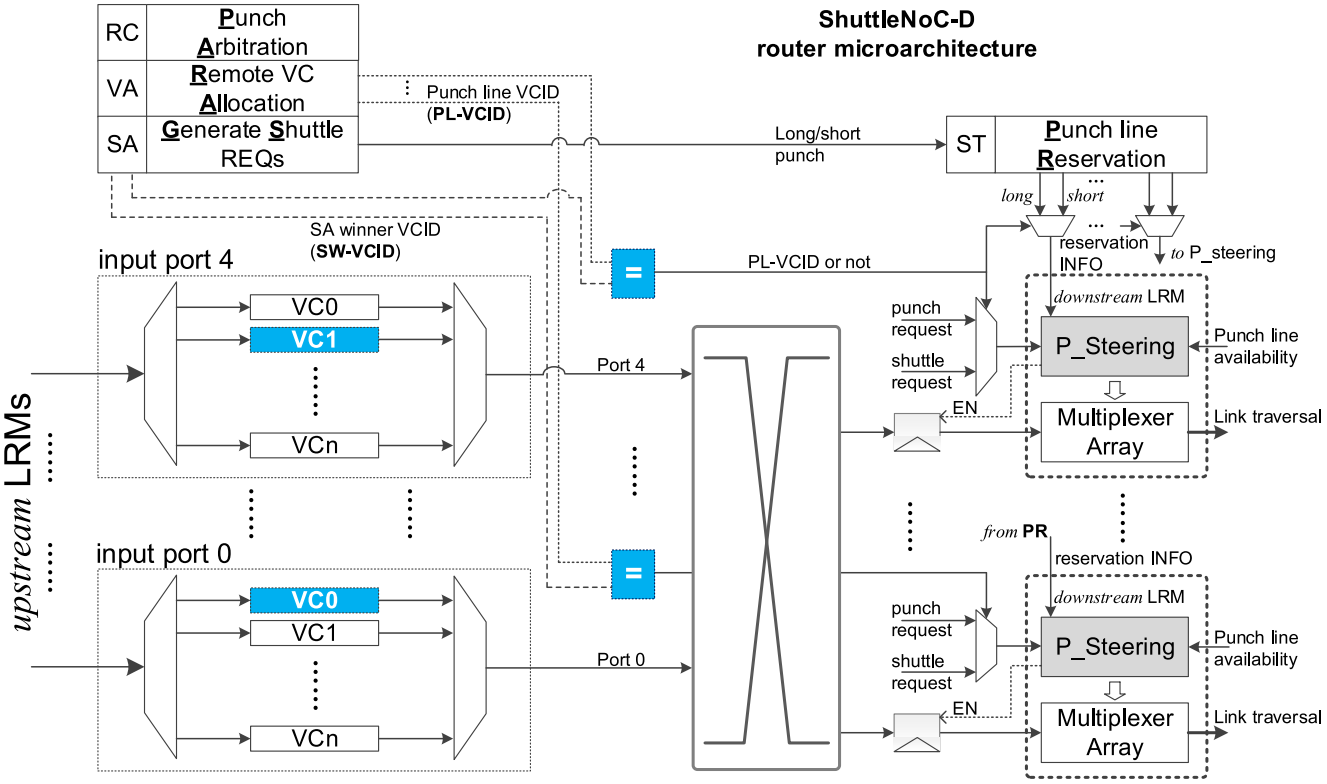


Fig. 8. ShuttleNoC-D router microarchitecture and the specific circuitry for the punch line reservation.

at each output port, because the arbitration in P_Steering module would only allow the winner to shuttle or punch into the downstream. The "EN" signal controls if the current port is the winner or not, so the flit must wait at the output latch if it fails the arbitration.

C. Power Adaptation Mechanisms

Previous sections have elaborated the hardware implementations of ShuttleNoC, including data paths on links as well as the router pipeline microarchitectures. In this section, we elaborate its power subsystem and how they collaborate with the ShuttleNoC microarchitecture to achieve localized power adaptation.

In order to support packet shuttling/punch, router microarchitecture must be capable of detecting the "on/off states" of the subrouters in the vicinity, as the references to propagate the corresponding shuttle or punch request to the LRM. Packet shuttling is implemented between neighboring subrouters, so their power states must be obtained in real time. Localized power adaptation also relies on runtime statistics as the reference to power on/off a subrouter in a particular subnetwork. In ShuttleNoC, three modules: 1) power metric computation (PMC); 2) local state ctrl (LSC); and 3) ctrl signal propagation (CSP) are added to serve these purposes and they consist of the low power subsystem in the router microarchitecture (Fig. 9), and they also apply to both instances of ShuttleNoC, specified as follows.

PMC: In order to quantify traffic intensity, we employ PMC module to compute the microarchitectural parameters at runtime. Previous work [5], [33]–[35] has proposed several

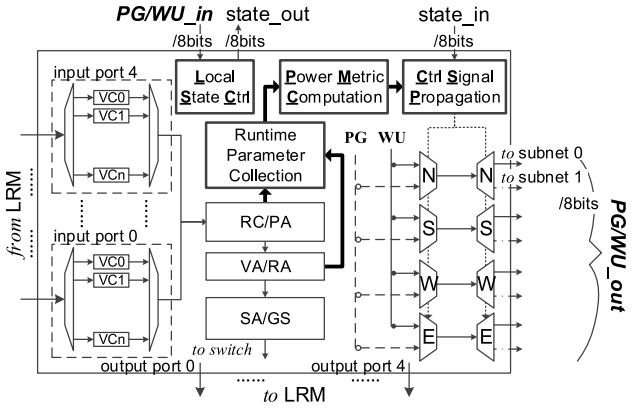


Fig. 9. ShuttleNoC power subsystem, which is used to collaborate with the hardware microarchitecture and fulfill the localized power adaptation.

reasonable congestion detection metrics, such as local injection queue occupancy, the average or maximum buffer occupancy, etc. These metrics can be seamlessly deployed in ShuttleNoC; however, to measure traffic intensity, the metric should be able to pinpoint the precise data path or direction that causes the packet contention. We then propose to use average flits queuing delay for each output direction as the intensity metric, formulated as $\mathrm{QD}_{\mathrm{outdir}} = (\sum_i^{N_{\mathrm{outdir}}} \mathrm{delay}_i / N_{\mathrm{outdir}})$, where N_{outdir} stands for the total number of flits heading direction "outdir." "delay_i" is the queuing delay that flit i must be retained in its input virtual channel, due to the failure of VA or SA. An increasing queueing delay may indicate that output virtual channels in outdir direction may be limited, and

Algorithm 1 Subnet Selection in CSP

Input: State of neighboring routers: *state in*; subnetworks: N;
PMC requests: *requests*;
Output: Subnet selected: n;
1: **for** each $req < dir, opa > \in request$ **do**
2: **if** $opa == PG$ **then**
3: **for** $(i = N - 1; i >= 0; i--)$ //shut down from the **highest-level do**
4: **if** $state\ in[dir][i] == WU$ **then**
5: return n; //power off subnet n at output dir
6: **end if**
7: **end for**
8: **else if** $opa == WU$ **then**
9: **for** $(i = 0; i<N; i++)$ //wake up from the **lowest-level do**
10: **if** $state\ in[dir][i] == PG$ **then**
11: return n; //wake up subnet n at output dir
12: **end if**
13: **end for**
14: **end if**
15: **end for**

a power adaptation request is supposed to be issued to higher level subrouter. These parameters can easily be obtained as soon as a packet has finished certain pipeline stages, without introducing additional overhead.

LSC: LSC is used to control the power state transition of its host subrouter. Bandwidth adaptation signals, namely PG/WU_in, are received from neighboring subrouters of four directions from its neighborhood. "PG" and "WU" indicate a "power-gated" or Wakeup request, respectively. If we use two subnets in ShuttleNoC, it is hence an 8-bit signal (2 subrouters×4 directions) and each bit is possible to be a PG or WU. By analyzing their numerical relationship, LSC decides to power on/off its host subrouter. This relationship is significant in determining power and performance tradeoff, and is also evaluated in Section IV.

CSP: Note that once PMC intends to issue a power adaptation request (PG/WU) to a certain output direction, CSP module is designed to inform the target subrouter, in that direction, of this request. state_in includes the on/off status sent from neighbor LSCs. CSP, based on this information, propagates power adaptation request by controlling the symmetrically organized MUXes, as Fig. 9 shows. Each bit of PG/WU_out will connect to the corresponding subrouter's LSC module in the neighborhood.

As specified above, CSP processes the power adaptation requests generated by PMC. It must decide which subrouter is supposed to be activated/deactivated based on their on/off status. Detailed procedure is shown in Algorithm 1. We stipulate the activation of subnets must be in order. For example, if subnet 1 is already active and subnets 2–4 are off, subnet 2 is then chosen as the activation candidate (lines 8–11). Shutting down, on the contrary, follows a reverse order by starting from the highest level active subnet (lines 2–5).

1) State Transition in the LSC: Clearly, the efficacy of power management mechanism depends on the dynamic router status, so similar to [5] and [23], we also use three power states to depict a subrouter: 1) Active; 2) Sleep; and 3) Wakeup. The state is maintained in LSC module. "Active" indicates the router is currently working and packet

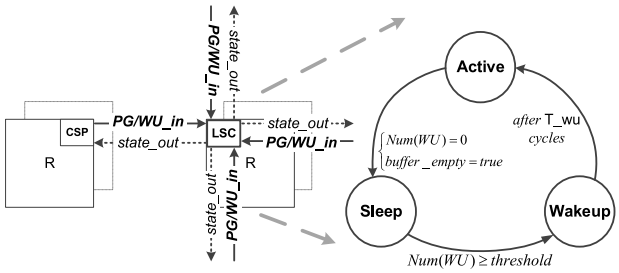


Fig. 10. LSC handshaking with CSP. LSC controls power state transitions of its host router, and we allocate three power states to be in align with the recently proposed schemes.

shuttling/punch is applicable, while Sleep and Wakeup means the router is power-gated or currently waking up, respectively. Packets are not allowed to shuttle or punch into Sleep and Wakeup routers. LSC and CSP are coupled as two sides of handshaking operation. Fig. 10 shows such interaction. Note that it only shows two neighboring subrouters of the same subnet, while LSC also interacts with CSPs that belong to other levels of subnets. For a particular subrouter, state transition from Active to Sleep must satisfy two conditions: 1) Num(WU) equals to 0, which denotes all bits in PG/WU_in are PGs and 2) no packet is remained in local input buffers. LSC can then safely shut down this router to reduce power consumption. By contrast, if LSC detects arbitrary PG/WU combinations in PG/WU_in, state transition from Sleep to Wakeup depends on if Num(WU) attains a predefined threshold. If so, it means the local bandwidth is more prone to be expanded. LSC will then wake up its host router. Once the router is waking up, it may take 10–20 cycles that active state will be finally attained, according to [5] and [23]. We use T_wu to indicate this transition delay in the figure.

IV. Evaluation

In this section, we evaluate the proposed ShuttleNoC architecture and the associate localized power adaptation mechanisms. First, we introduce the platform and baselines we use. Second, we show various results and discussions in terms of the performance and runtime power efficiency.

A. Experimental Setup

1) Platform: We evaluate ShuttleNoC at the network and full system level.

1) From the network perspective, we modified Booksim2.0 [36] simulator to run application traces, which is extracted from GEM5 [37], a full-system cycle-accurate simulator. The fundamental NoC topology includes 4×4 and 8×8 mesh. We also construct different scales of mesh/torus topology in the sensitivity analysis. On-chip router is configured with a four-stage pipeline plus one cycle for shuttle or punch. We use four virtual channels for the input buffer with 5-flit depth each.
2) From the full-system perspective, we carry out the cycle-accurate simulation using GEM5 running selected PARSEC [38] benchmarks. The basic platform contains

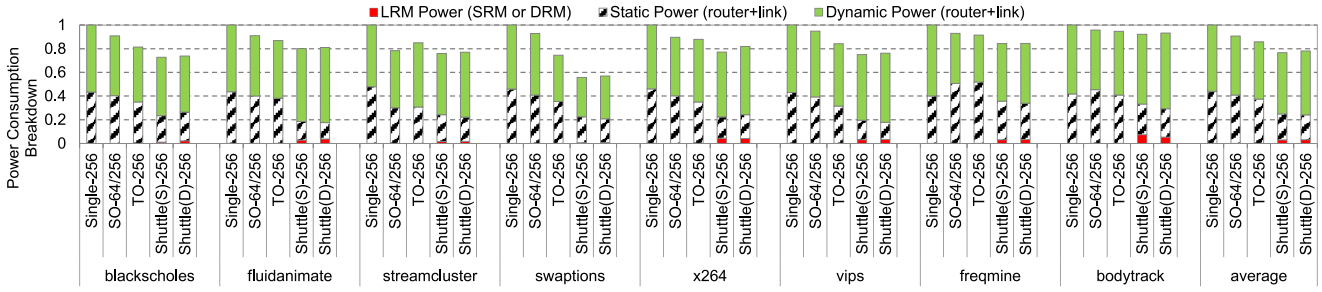


Fig. 11. Power consumption breakdown (normalized to single-256) for ShuttleNoC (S/D) and the baselines. We analyze dynamic and static power as well as ShuttleNoC specific LRM power.

16 alpha cores with private L1 (32KB, 2-way) and shared L2 (256KB, 2-way, MESI) caches. This configuration is also used for extracting the application traces for network-only evaluations.

Both computation and communication infrastructures are clocked at 1.16V/0.8 GHz for power estimation. This *V*/*F* setting is also aligned with the Intel single-chip cloud computer (SCC) [39] under 45-nm technology node. We extract component statistics from GEM5 and use McPAT [40] with CACTI for the computation power estimation, and DSENT [41] power model for the NoC. ShuttleNoC does not apply DVFS to the router and links so it does not incur any cross-domain synchronization as in the SCC or previous proposed techniques [19]. Besides, it is also unnecessary to worry about the misalignment of core-NoC frequencies under such configuration. We also employ Synopsis Design Compiler [42] to obtain the area overhead of various NoC designs under SMIC90 technology library.

2) Baselines: We use three baselines to prove the efficacy of ShuttleNoC in the power adaptation.

1) The first one is a traditional single NoC with no power management involved. We configure its bandwidth as 256 bits, so the first baseline is referred to as "single-256."
2) The second baseline, referred to as "SO-64/256," is the "spatial-oriented" approach, and two bandwidth configurations are set as 64 bits and 256 bits as in [15].
3) The third one, referred to as "TO-256," is the "temporal-oriented" approach [5] with four subnets and each one 64 bits wide.

The same configuration is also set for the proposed ShuttleNoC. Besides, T_wu is set as 20 cycles for a waking-up subrouter to finally attain Active state [5] [23]. Since we use four subnets, state_in/out and PG/WU_out/in are both 16 bits (4 subrouters $\times$ 4 directions) for the two instances of ShuttleNoC. Note that even if the bandwidth configuration for the baselines and ShuttleNoC is not exactly the same, the maximum bandwidth (256 bits) is strictly equal. ShuttleNoC does not intentionally broaden the bandwidth in our evaluations.

Theoretically, punch packet in ShuttleNoC-D is able to pass a series of nodes, as long as the punch lines along its path is available. However, we only restrain the distance of *long* punch within three nodes and *short* punch within two nodes, because from the implementation point of view, prolonged wires will elongate the packet traverse cycles and the NoC clock frequency must be limited within a certain range. To make the simulation more practical, we do not allow much longer punch. However, in the chip design phase, this parameter could be dynamically scaled based on the desired operating voltage and frequency settings. The proposed packet steering and power adaptation mechanism are kept unchanged.

B. Results and Analysis

1) Network-Level Power and Performance Tradeoff: ShuttleNoC enables the localized power adaptation, so in this set of experiment, we show how much benefit it brings in terms of overall power reduction in 8×8 mesh. Fig. 11 plots the decomposition of NoC power to examine the impact of each component. Compared with single-256, SO-64/256, and TO-256, ShuttleNoC-S shows substantial total power reduction by 26.3%, 17.1%, and 12.1%, while ShuttleNoC-D shows an improvement of 24.9%, 15.7%, and 10.7%, slightly less beneficial than ShuttleNoC-S.

Discussion: The improvement comes from the localized power adaptation, mainly from the reduction of static power consumption. ShuttleNoC only activates/deactivates subrouter based on the local traffic intensity, rather than fixing the bandwidth as done in SO-64/256. Compared to TO-256, it does not need to affect neighboring node to maintain the network connectivity, so static power is also reduced. On the other hand, ShuttleNoC employs LRM to achieve packet shuttling/punch, so dynamic power may increase due to the frequent link reconfigure operation. As shown in the figure, it incurs 3.8% power overhead due to the SRMs, and 1.7% and 3.3% dynamic power increase compared to SO-64/256 and TO-256, respectively. However, the abundant static power reduction still guarantees the two instances of ShuttleNoC an overall power saving compared with the baselines.

To further prove the effectiveness of ShuttleNoC, we then show the performance degradation to explore the power and performance tradeoff. Average packet latency is used as the performance metric. The result in Fig. 12 shows that ShuttleNoC-S degrades average network performance by 9.6%, and 5.0% for ShuttleNoC-D, compared with single-256. While, SO-64/256 and TO-256 demonstrate more severe performance degradation as 14.1% and 27.2%, respectively.

Discussion: One may suspect that why power reduction does not introduce severe network performance degradation

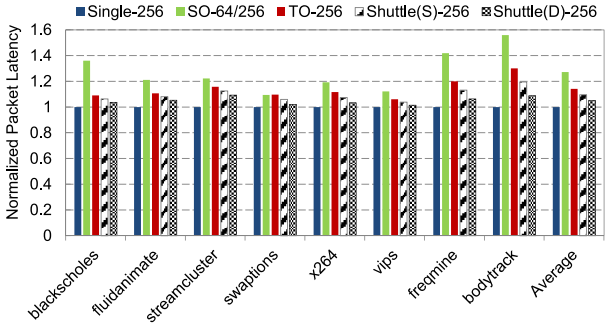


Fig. 12. Performance comparison, illustrated via average packet latency, between ShuttleNoC and the baselines.

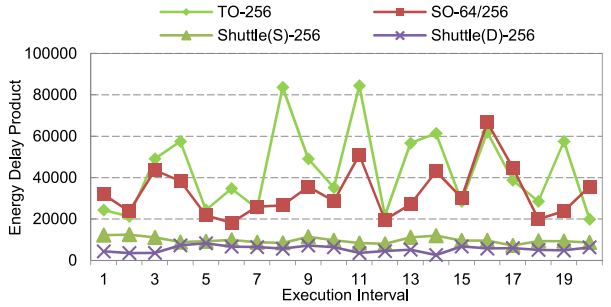


Fig. 13. ShuttleNoC runtime EDP. This metric is used to convey power efficiency.

for ShuttleNoC architectures. The reason is that although it limits the bandwidth resources, it provides more flexible link connectivity so the head-of-line blocking problem is effectively alleviated. Packets have more path candidates to proceed according to the location of the destination, instead of waiting for previous queued packets to be first processed. This direct benefit is not attainable in conventional one-case-for-all or spatial-oriented approach. Temporal-oriented approach like TO-256 baseline could, to some extent, alleviate head-of-line-blocking, but only at network injection phase. ShuttleNoC could serve in-transit packet at any location, so the latency increment brought by bandwidth reduction is overshadowed by the improvement brought by data path flexibility. ShuttleNoC-D provides additional punch lines for latency-sensitive traffics that further contributes to the overall performance improvement.

As a further proof, we evaluate the power efficiency variation periodically, by executing one of our selected benchmarks (swaptions). We use energy delay product (EDP) as the power efficiency representative. In the snapshot shown in Fig. 13, we observe that EDP values remain almost constant: 5.1% variation on average for ShuttleNoC-S and 4.8% for ShuttleNoC-D. Whereas, TO-256 exhibits large fluctuations during execution, because of the coarse-grained, subnet-level power control. For SO-64/256, it suffers from the severe performance degradation when heavy traffic migrates to small routers. Similar behavior is also observed in other benchmarks and other snapshots within the same benchmark.

2) Full-System Power and Performance: We intend to explore the fraction of the ShuttleNoC power reduction at the system level. We present the full system power and the breakdown of the cores and NoC power, respectively, in Fig. 14. For the selected 8 PARSEC benchmarks, the average fraction of NoC power over the total chip power is 17.7% and 17.8% under the preset chip-level voltage and frequency settings. The average power reduction is 6.8% and 5.9% compared with Single-256. These results coincide with the numbers reported in Sections I and IV-B1. The fraction of NoC over total chip power is nearly 18%, so the benefit of ShuttleNoC in optimizing system power is around 7%. Similarly, system power reduction does not incur obvious performance loss. Table I shows the detailed performance degradation results obtained from our cycle-accurate simulation platform. Compared with the original classic singular-width NoC design, performance degradation is graceful with only 1.4% and 0.8% decrement for the two representatives of ShuttleNoC, and that proves the power savings at the communication infrastructure does not introduce severe system performance loss.

Note that in full system evaluation, we does not apply any power management schemes such as DVFS on computation resources, so the power results of cores remain nearly the same (slightly different) for the ShuttleNoC and the employed baseline. We only want to show the impact of ShuttleNoC to the system performance and power, while DVFS for cores could also be applied seamlessly in accompany with the ShuttleNoC if more aggressive power/performance tradeoff is required at the chip level.

3) Heterogeneity Adaptation Analysis: The smooth power efficiency of ShuttleNoC stems from the effective adaptation of traffic heterogeneity, the most important feature of ShuttleNoC. As evidence, this set of experiment traces the latency variation of each node in a 4×4 NoC, by executing the same benchmark in the previous experiment. This time, we calculate the average packet latency for the packets that terminates at this node at an interval of 1 ms. As shown in Fig. 15, SO-64/256 exhibits obvious latency variations between 40–190 cycles. Due to the large bandwidth of big routers in the center, nodes 6, 7, 10, and 11 show a moderate latency variation compared to nodes at boundaries like 1 or 15, but still around 100 cycles. TO-256 shows a mild latency variation around 110 cycles on average. By sharp contrast, ShuttleNoC-S shows a more smooth latency variation around 30–40 cycles, and ShuttleNoC-D around 20–30 cycles due to the contribution of punch lines. Such near-constant latency further proves that ShuttleNoC has the unique ability of localized bandwidth adaptation, and thus more possible to achieve optimal power efficiency.

4) Sensitivity Analysis:

a) Request threshold and ShuttleNoC responsiveness: In this experiment, we first evaluate the impact of the Wakeup request threshold—Num(WU) in Fig. 10, to the ShuttleNoC responsiveness. We employ ShuttleNoC-S to evaluate this key parameter. Speaking of the responsiveness, we evaluate the NoC power consumption and average packet latency, with Num(WU) tuned from the minimum to the maximum. We evaluate all the selected benchmarks as the design space exploration, seeking to explore the tradeoff between the network performance and power imposed by this parameter. As can

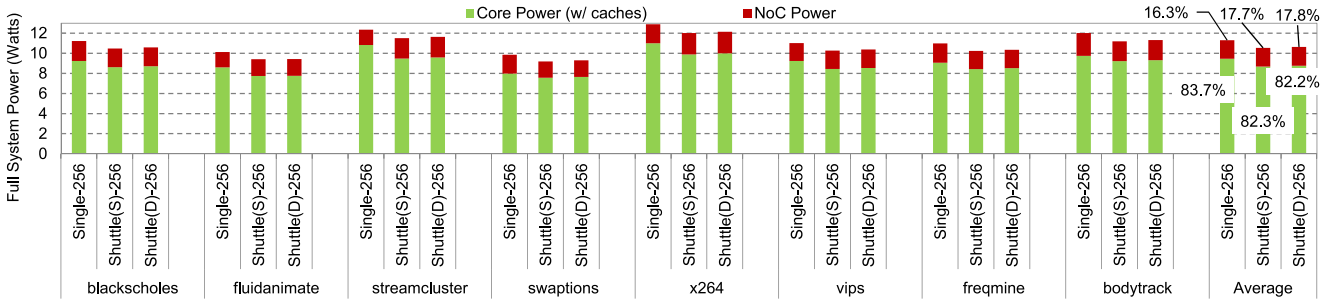


Fig. 14. Full system power breakdown.

TABLE I
FULL-SYSTEM EXECUTION TIME (IN CYCLES) ANALYSIS. THE PERFORMANCE DEGRADATION PERCENTAGE IS MARKED IN BOLD

	Black.		Flui..		Stream.		Swap.		x264		vips		Freqm.		Bodyt.	
Sing.-256.	$1.863x10^8$	N/A	$2.302x10^9$	N/A	$2.582x10^9$	N/A	$1.724x10^9$	N/A	$3.539x10^9$	N/A	$2.286x10^9$	N/A	$3.952x10^9$	N/A	$2.529x10^9$	N/A
Shuttle(S)	$1.886x10^8$	**1.23%**	$2.339x10^9$	**1.60%**	$2.613x10^9$	**1.20%**	$1.759x10^9$	**2.03%**	$3.596x10^9$	**1.61%**	$2.322x10^9$	**1.57%**	$3.979x10^9$	**0.68%**	$2.569x10^9$	**1.58%**
Shuttle(D)	$1.869x10^8$	**0.32%**	$2.328x10^9$	**1.12%**	$2.602x10^9$	**0.77%**	$1.752x10^9$	**1.62%**	$3.571x10^9$	**0.90%**	$2.341x10^9$	**1.22%**	$3.964x10^9$	**0.30%**	$2.548x10^9$	**0.75%**

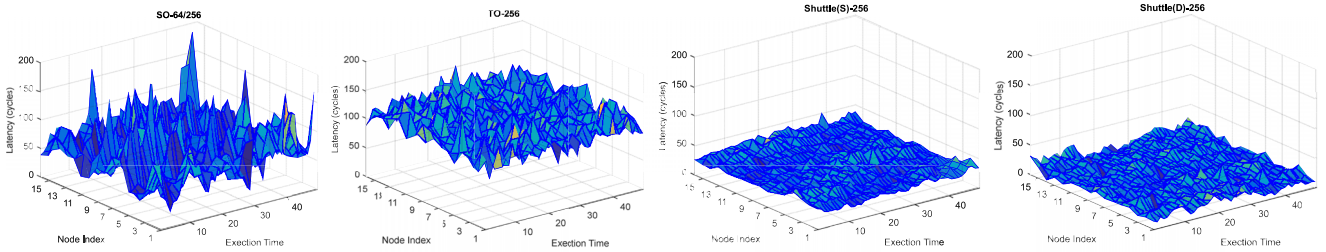


Fig. 15. Heterogeneity adaptation analysis. We use surfplot to visualize the fluctuation of latency stats at each node. ShuttleNoC is more streamlined compared with the other baselines.

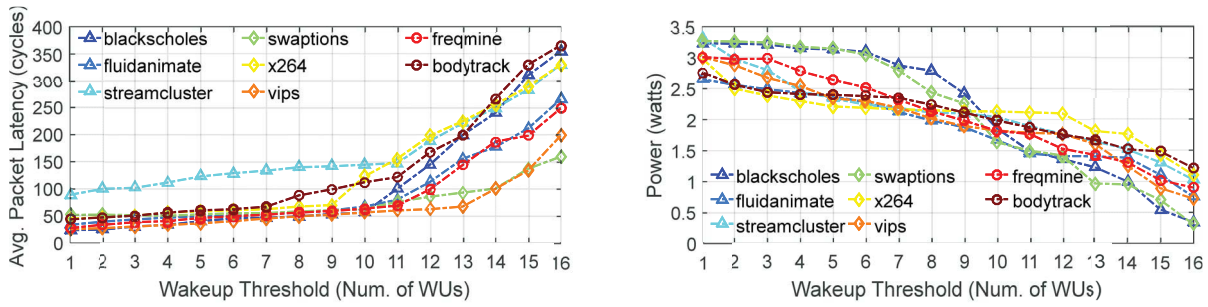


Fig. 16. Sensitivity analysis of the router wakeup threshold—Num(WU), versus the NoC power consumption.

be seen from the "hockey" curves in Fig. 16, the workloads demonstrate a uniform behavior. Taking swaptions as an example, the packet latency remains almost stable with threshold varied from 1 to 10, while the power reduces nearly 50%. This phenomenon proves that larger threshold contributes to the power reduction due to the prolonged subrouter sleeping cycles without compromising performance. However, the latency starts to climb significantly to 160 cycles (207.7% degradation) from 10 to 16. NoC performance is suffered under this scenario, because subrouters become too "lazy" to respond to the increasing WU requests, even if the total power continues to decrease. Different benchmarks have different inflexion point in its hockey curve, almost around 8–10. Therefore, in the previous experiments, we all set the request threshold as 10 to acquire a better energy/performance tradeoff. Actually, ShuttleNoC can actually work at other threshold values, contingent to the available power budget at the chip level or the intended NoC power efficiency.

b) Subnet configuration: As can be imagined, different scales of subnets could influence the overall network performance. That is, because the subnet configuration affects the bandwidth and further determines the packet size of network traffic. Our baseline single-256 could be regarded as an extreme case that contains only one subnet with 256 bits bandwidth. Fig. 17 also explores the 2-subnet configuration with 128 bits each to evaluate the performance variation. We

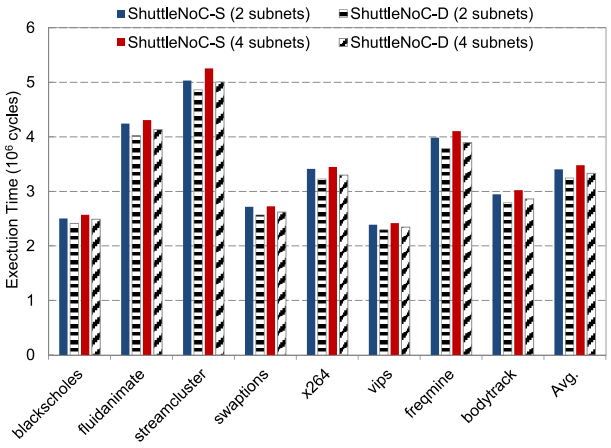


Fig. 17. Sensitivity analysis of the subnet configuration. We use total trace execution cycles as the performance metric.

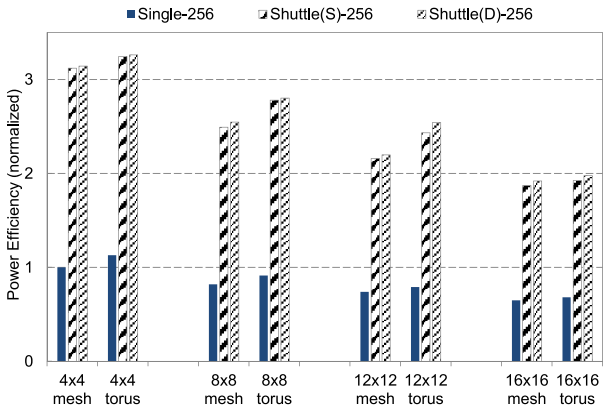


Fig. 18. Topology scaling and its impact to power efficiency. We evaluate different scales of mesh and torus topology, ranging from 4×4 to 16×16. The higher the better.

use total execution cycles as the metric this time and analyze the results from two perspectives. First, comparing to the 2-subnet configuration, the 4-subnet configuration yields 2.28% and 2.61% higher average execution cycles. It proves the common sense that large bandwidth generates higher performance. Similar result is also observed in Fig. 12 for the extreme case Single-256. However, the power consumption is not that optimistic under larger bandwidth. Oracle configuration should reference the power efficiency constraint and set the number of subnets accordingly. Second, within the range of 2-subnet configuration, ShuttleNoC-D still outperforms ShuttleNoC-S, which is aligned with the 4-subnet configuration.

c) Topology settings: We evaluate the topology scaling ranging from 4×4 to 16×16 mesh and torus topologies, and explore the power efficiency under these cases running the benchmark trace of swaptions. We calculate the reciprocal of EDP as the power efficiency metric as shown in Fig. 18. The results are normalized to the 4×4 mesh. The behavior of topology scaling is uniform, larger scale has lower power efficiency because the traffic volume also increases that causes an enlarged packet latency. For torus, we does not allocate punch lines on the end-to-end loopback wires, and sets the latency of the loopback wires as three cycles, identical to the punch line latency. Even so, torus still has a slightly higher performance due to the shortcut of loopback wires. It is worth mention that ShuttleNoC is a general design concept that could be applied on any other topology besides mesh and torus.

d) Punch line usage: The two design parameters α and β are the incoming packet is endorsed to use the local punch line or not. Intuitively, it is tricky to select the two values because mishandling the proportion of shuttle and punch packets will form the resource competition. In specific, α too small will shrink the amount of shuttle packets and further intensify the competition for the punch lines. As another example, β too large will shrink the amount of long punches, so rush packets can only acquire inadequate accelerations. Our design exploration proves this notion. Fig. 19 shows four benchmarks with different computation and communication intensities under α/β scaling. They all exhibit a valley area in the near central of the mesh plot. Average packet latency is

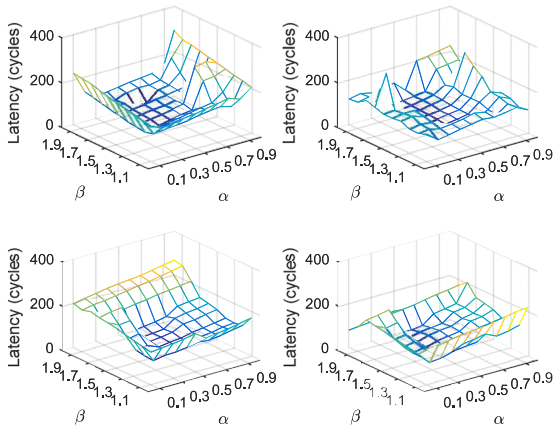


Fig. 19. α and β scaling and its impact to average packet latency (in cycles). We use mesh plot to clearly demonstrate the "valley" in the central area. The four benchmarks are blackscholes (top-left), $\times$264 (top-right), freqmine (bottom-left), and fluidanimate (bottom-right).

TABLE II
AREA OVERHEAD BREAKDOWN (mm^2). THE TOTAL AREA IS NORMALIZED TO SINGLE-256

ITEM	Buffer	Crossbar	Control Logic	Clock	Link	LRM	LSC +PMC +CSP	Total	Overhead
Single-256	4.174	0.614	0.068	0.187	3.301	N/A	N/A	8.344	1 (baseline)
TO-4x64	4.161	0.407	0.384	0.582	3.729	N/A	N/A	9.263	**1.110x**
Shuttle(S)-4x64	4.169	0.409	0.691	0.582	3.836	0.384	0.23	10.301	**1.235x**
Shuttle(S)-2x64	4.167	0.356	0.594	0.298	3.524	0.192	0.128	9.259	**1.109x**
Shuttle(D)-4x64	4.374	0.408	0.748	0.583	4.649	0.421	0.234	11.417	**1.368x**
Shuttle(D)-2x64	4.387	0.362	0.649	0.294	3.658	0.216	0.132	9.698	**1.162x**

minimum in this area. Whereas in the mesh boundaries, the execution cycles exhibit obvious upward. Therefore, in practical use, we need to carefully select the two values. A viable option is to make them on-line tunable to accommodate different executing phases, while local coordination at each NoC node is hardly to ensure a global optimal performance. We still need an extra central monitor, beside the many-core chip, and do global optimization that will inevitably increase the implementation cost. We will leave this problem as our future work.

e) Overhead analysis: ShuttleNoC relies on LRM and dedicated hardware in routers to fulfill the power adaptation purpose. We evaluate the implementation cost of ShuttleNoC by comparing its area overhead to other NoC designs, as shown in Table II. We analyze the area contributive components. ShuttleNoC incurs a mild increase of the total NoC area, due to the complicated link layout and the dedicated control modules. Fortunately, LRM only consists of several multiplexers, and does not introduce significant area overhead (up to 3.7%). Compared with Single-256, the area increase is up to 1.37× in total. Note that although ShuttleNoC exhibits more area overhead, it does not consume a larger power, because the proposed power adaption mechanism brings more power savings, and even gives a better play in power/performance tradeoff.

V. Conclusion

This paper proposes ShuttleNoC, a novel NoC architecture to enforce optimal power efficiency for the communication infrastructure in manycores. Unlike previous temporal and spatial-oriented approach, ShuttleNoC achieves localized power adaptation to serve runtime traffic heterogeneity. By designing precise power state transition mechanism and dedicated LRM, packets are allowed to shuttle/punch between multiple subnetworks. Compared with temporal-oriented approach, ShuttleNoC avoids unnecessary activation of subrouters to obtain lower power consumption. Besides, ShuttleNoC does not resort to fixed bandwidth configurations as in spatial-oriented approach, and hence yields higher power efficiency. We also demonstrate two representatives under ShuttleNoC design paradigm, that is, ShuttleNoC-S and ShuttleNoC-D, elaborating their pros and cons and respective hardware implementation in link and router microarchitecture. At last, we specify how the associate power adaptation mechanisms are supported in the two representatives for localized power adaptation. We believe that ShuttleNoC is a promising architecture to achieve optimal power efficiency for the communication infrastructure in future many-core processors.

References

[1] S. Borkar, "Networks for multi-core chips," in *Proc. Special Session ACM/IEEE Int. Symp. Low Power Electron. Design*, 2007, pp. 1–6.

[2] "Niagara 2 opens the floodgates," Sun Microsyst. Inc., Santa Clara, CA, USA, Rep., Nov. 2006.

[3] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sep./Oct. 2007.

[4] C. Rajamanikkam, J. S. Rajesh, K. Chakraborty, and S. Roy, "BoostNoC: Power efficient network-on-chip architecture for near threshold computing," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, 2016, pp. 1–8.

[5] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, "Catnap: Energy proportional multiple network-on-chip," presented at the 40th Ann. Int. Symp. Comput. Archit., Tel Aviv, Israel, 2013.

[6] J. Zhan, Y. Xie, and G. Sun, "NoC-sprinting: Interconnect for fine-grained sprinting in the dark silicon era," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2014, pp. 1–6.

[7] A. Sharifi, A. K. Mishra, S. Srikantaiah, M. Kandemir, and C. R. Das, "PEPON: Performance-aware hierarchical power budgeting for NoC based multicores," presented at the 21st Int. Conf. Parallel Archit. Compilation Techn., Minneapolis, MN, USA, 2012.

[8] H. Bokhari, H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran, "darkNoC: Designing energy-efficient network-on-chip with multi-Vt cells for dark silicon," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2014, pp. 1–6.

[9] R. Parikh, R. Das, and V. Bertacco, "Power-aware NoCs through routing and topology reconfiguration," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2014, pp. 1–6.

[10] K. Swaminathan *et al.*, "Steep-slope devices: From dark to dim silicon," *IEEE Micro*, vol. 33, no. 5, pp. 50–59, Sep./Oct. 2013.

[11] H. Lu, G. Yan, Y. Han, Y. Wang, and X. Li, "ShuttleNoC: Boosting on-chip communication efficiency by enabling localized power adaptation," in *Proc. 20th Asia South Pac. Design Autom. Conf. (ASP-DAC)*, 2015, pp. 142–147.

[12] A. Samih *et al.*, "Energy-efficient interconnect via router parking," in *Proc. IEEE 19th Int. Symp. High Perform. Comput. Archit. (HPCA2013)*, 2013, pp. 508–519.

[13] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[14] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," presented at the 37th Annu. Int. Symp. Comput. Archit., St.-Malo, France, 2010.

[15] A. K. Mishra, N. Vijaykrishnan, and C. R. Das, "A case for heterogeneous on-chip interconnects for CMPs," in *Proc. 38th Annu. Int. Symp. Comput. Archit. (ISCA)*, San Jose, CA, USA, 2011, pp. 389–399.

[16] S. Ma, N. E. Jerger, Z. Wang, M. Lai, and L. Huang, "Holistic routing algorithm design to support workload consolidation in NoCs," *IEEE Trans. Comput.*, vol. 63, no. 3, pp. 529–542, Mar. 2014.

[17] M. Sheng, N. E. Jerger, and Z. Wang, "DBAR: An efficient routing algorithm to support multiple concurrent applications in networks-on-chip," in *Proc. 38th Annu. Int. Symp. Comput. Archit. (ISCA)*, San Jose, CA, USA, 2011, pp. 413–424.

[18] R. Hesse, J. Nicholls, and N. E. Jerger, "Fine-grained bandwidth adaptivity in networks-on-chip using bidirectional channels," in *Proc. IEEE/ACM 6th Int. Symp. Netw. Chip*, 2012, pp. 132–141.

[19] A. K. Mishra *et al.*, "A case for dynamic frequency tuning in on-chip networks," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, New York, NY, USA, 2009, pp. 292–303.

[20] H. Lu, G. Yan, Y. Han, and X. Li, "PowerTrader: Enforcing autonomous power management for future large-scale many-core processors," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 3, no. 4, pp. 283–295, Oct./Dec. 2017.

[21] K. Ma, X. Li, M. Chen, and X. Wang, "Scalable power control for many-core architectures running multi-threaded applications," presented at the 38th Annu. Int. Symp. Comput. Archit., San Jose, CA, USA, 2011.

[22] H. Matsutani, M. Koibuchi, H. Amano, and D. Wang, "Run-time power gating of on-chip routers using look-ahead routing," in *Proc. Asia South Pac. Design Autom. Conf. (ASPDAC)*, 2008, pp. 55–60.

[23] L. Chen and T. M. Pinkston, "NoRD: Node-router decoupling for effective power-gating of on-chip routers," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Vancouver, BC, Canada, 2012, pp. 270–281.

[24] L. Chen, D. Zhu, M. Pedram, and T. M. Pinkston, "Power punch: Towards non-blocking power-gating of NoC routers," in *Proc. IEEE 21st Int. Symp. High Perform. Comput. Archit. (HPCA)*, Burlingame, CA, USA, 2015, pp. 378–389.

[25] P. Gratz *et al.*, "On-chip interconnection networks of the TRIPS chip," *IEEE Micro*, vol. 27, no. 5, pp. 41–50, Sep./Oct. 2007.

[26] D. Wentzlaff *et al.*, "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, Sep./Oct. 2007.

[27] N. E. Jerger, A. Kannan, Z. Li, and G. H. Loh, "NoC architectures for silicon interposer systems: Why pay for more wires when you can get them (from your interposer) for free?" in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2014, pp. 458–470.

[28] S. Volos *et al.*, "CCNoC: Specializing on-chip interconnects for energy efficiency in cache-coherent servers," in *Proc. IEEE/ACM 6th Int. Symp. Netw. Chip*, 2012, pp. 67–74.

[29] A. K. Mishra, O. Mutlu, and C. R. Das, "A heterogeneous multiple network-on-chip design: An application-aware approach," in *Proc. 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Austin, TX, USA, 2013, pp. 1–10.

[30] A. Flores, J. L. Aragon, and M. E. Acacio, "Heterogeneous interconnects for energy-efficient message management in CMPs," *IEEE Trans. Comput.*, vol. 59, no. 1, pp. 16–28, Jan. 2010.

[31] M. Ramakrishna, P. V. Gratz, and A. Sprintson, "GCA: Global congestion awareness for load balance in networks-on-chip," in *Proc. 7th IEEE/ACM Int. Symp. Netw. Chip (NoCS)*, 2013, pp. 1–8.

[32] F. A. Samman, T. Hollstein, and M. Glesner, "Runtime contention and bandwidth-aware adaptive routing selection strategies for networks-on-chip," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1411–1421, Jul. 2013.
[33] Y. S.-C. Huang, K. C.-K. Chou, C.-T. King, and S.-Y. Tseng, "NTPT: On the end-to-end traffic prediction in the on-chip networks," in *Proc. 47th ACM/IEEE Design Autom. Conf. (DAC)*, Anaheim, CA, USA, 2010, pp. 449–452.
[34] H. Sasaki, S. Imamura, and K. Inoue, "Coordinated power-performance optimization in manycores," in *Proc. 22nd Int. Conf. Parallel Archit. Compilation Techn. (PACT)*, 2013, pp. 51–61.
[35] U. Y. Ogras and R. Marculescu, "Prediction-based flow control for network-on-chip traffic," in *Proc. 43rd ACM/IEEE Design Autom. Conf.*, San Francisco, CA, USA, 2006, pp. 839–844.
[36] *Booksim2.0*. Accessed: Jun. 12, 2017. [Online]. Available: https://nocs.stanford.edu/
[37] (2018). *The Gem5 Simulator*. [Online]. Available: http://www.gem5.org/Main_Page
[38] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proc. Int. Conf. Parallel Archit. Compilation Techn.*, 2008, pp. 72–81.
[39] P. Salihundam *et al.*, "A 2 Tb/s 6 × 4 mesh network for a single-chip cloud computer with DVFS in 45 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 46, no. 4, pp. 757–766, Apr. 2011.
[40] S. Li *et al.*, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, New York, NY, USA, 2009, pp. 469–480.
[41] C. Sun *et al.*, "DSENT—A tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *Proc. IEEE/ACM 6th Int. Symp. Netw. Chip*, 2012, pp. 201–210.
[42] Synopsis. *Design Compiler*. Accessed: Mar. 2, 2015. [Online]. Available: http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DesignCompiler/Pages/default.aspx

Hang Lu received the Ph.D. degree from the University of Chinese Academy of Sciences, Beijing, China, in 2015.

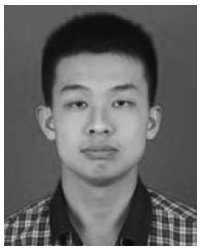
He is currently an Assistant Professor with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His current research interests include high performance networks-on-chip, power efficient many-core architectures, scale-out processors, and high performance neural network accelerators.

Yisong Chang received the B.S., M.S., and Ph.D. degrees in computer science and technology from Tianjin University, Tianjin, China, in 2008, 2010, and 2013, respectively.

He is currently an Assistant Professor with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. His current research interests include heterogeneous accelerators, memory subsystem, and system-level HW-SW co-design.

Guihai Yan (M'11) received the B.Sc. degree in electronics and software engineering (dual-degree) from Peking University, Beijing, China, in 2005 and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2011.

He is currently an Associate Professor with the Institute of Computing Technology, Chinese Academy of Sciences. His current research interests include high performance computer architecture, domain-specific microsystems, and energy-efficient computing.



Ning Lin received the B.Eng. degree from Xiangtan University, Xiangtan, China, in 2016. He is currently pursing the Ph.D. degree with the University of Chinese Academy of Sciences, Beijing, China.

His current research interests include computer architecture, low-power design of deep learning algorithms, DNN compression, and DNN acceleration on mobile and embedded devices.

Xin Wei received the B.Eng. degree in electronic information science and technology from Zhengzhou University, Zhengzhou, China, in 2016. He is currently pursing the master's degree with the University of Chinese Academy of Sciences, Beijing, China.

His current research interests include convolutional neural network accelerator on embedded devices and convolution neural network algorithm optimization.

Xiaowei Li (SM'04) received the B.Eng. and M.Eng. degrees in computer science from the Hefei University of Technology, Hefei, China, in 1985 and 1988, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 1991.

He was an Associate Professor with the Department of Computer Science and Technology, Peking University, Beijing, from 1991 to 2000. In 2000, he joined ICT, CAS, as a Professor, where he is currently the Deputy Director of the State Key Laboratory of Computer Architecture. He has co-authored over 280 papers in journals and international conferences, and he holds 60 patents and 30 software copyrights. His current research interests include very large scale integration testing, design for testability, design verification, dependable computing, and wireless sensor networks.

Dr. Li has been the Vice Chair of the IEEE Asia & Pacific Regional Test Technology Technical Council since 2004, where he is currently Vice Chair. He was the Chair of the Technical Committee on Fault Tolerant Computing, China Computer Federation from 2008 to 2012, and the Steering Committee Chair of IEEE Asian Test Symposium from 2011 to 2013. He was the Steering Committee Chair of IEEE Workshop on RTL and High Level Testing from 2007 to 2010. He services as an Associate Editor of the *Journal of Computer Science and Technology*, the *Journal of Low Power Electronics*, the *Journal of Electronic Testing: Theory and Applications*, and the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.