Surf-Bless: A Confined-interference Routing for Energy-Efficient Communication in NoCs

Peng Wang[†‡*], Sobhan Niknam[†], Sheng Ma[‡*], Zhiying Wang[‡], Todor Stefanov[†*]
[†]Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands
[‡]State Key Laboratory of High Performance Computing, National University of Defense Technology, China
[†]{p.wang, s.niknam, t.p.stefanov}@liacs.leidenuniv.nl, [‡]{mashnudt, zywang}@nudt.edu.cn

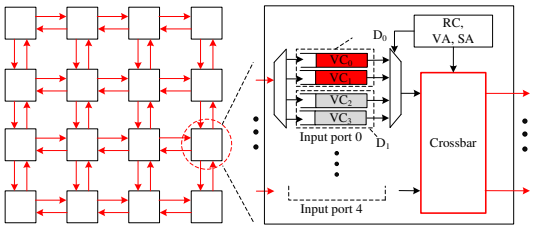
ABSTRACT

In this paper, we address the problem of how to achieve energy-efficient confined-interference communication on a bufferless NoC taking advantage of the low power consumption of such NoC. We propose a novel routing approach called Surfing on a Bufferless NoC (Surf-Bless) where packets are assigned to domains and Surf-Bless guarantees that interference between packets is confined within a domain, i.e., there is no interference between packets assigned to different domains. By experiments, we show that our Surf-Bless routing approach is effective in supporting confined-interference communication and consumes much less energy than the related approaches.

1 INTRODUCTION

A Network-on-Chip (NoC) with low latency, high bandwidth, and good scalability is a promising communication infrastructure for large size many-core systems. However, as the NoC resource are shared by different packets, there may be significant interference between packets, which influences packet transmission time. If the packets come from different applications, the temporal behaviour of an application depends on the behavior of other applications, thereby making a many-core system non-composable. In a composable system, applications are completely isolated and cannot affect each others functional or temporal behaviors [1]. One way to remove the applications dependency caused by the interference between packets, a necessary step to make the system composable, is to group the packets of different applications into different domains and keep non-interference between domains [2, 3]. Thus, the packets in one domain have no influence on the transmission time of the packets in other domains. Such packet transmission scheme we call confined-interference communication.

The interference between packets is caused by the contention on the shared resource of a NoC, such as virtual channels (VCs), crossbars, input/output ports, and links. In order to support confined-interference communication, these shared resources should be separated in space or in time. A straightforward way to implement confined-interference communication is to assign different VCs to hold packets of different domains (isolation in space) and to split the utilization time of input/output ports, crossbars and links into multiple time slots, then assign different time slots to different domains at design-time (isolation in time). For example, as shown in Figure 1, the NoC is used to transfer packets of two domains (D_0 and D_1). The VCs (VC_0, VC_1, VC_2, and VC_3) are assigned to different domains. VC_0 and VC_1 are assigned to D_0. VC_2 and VC_3 are assigned to D_1. Packets of D_0 and D_1 can be held only in their own VCs. At even clock cycles, only packets of D_0 can go through the crossbars, output ports, and links (distinguished by the red color in Figure 1(a) and 1(b)). At odd clock cycles, only packets of D_1 can go through the crossbars, output ports, and links. In this way, there is no interference between D_0 and D_1 because packets of different domains are completely isolated in space and time. However, to separate the packets in space, each domain requires at least one VC, which needs a large number of buffers and causes high static and dynamic power consumption [4, 5]. For example, the NoC with two 16-flit VCs per input port in Teraflop [6] and the NoC with four 1-flit VCs and two 3-flit VCs per input port in Scorpio [7] consume up to 28% and 19% of the total system power, respectively. In fact, such high power consumption of a NoC has become the major bottleneck that prevents applying NoCs on many-core systems [8]. Therefore, it is critical to implement confined-interference communication with low power/energy consumption.



(a) Link time-multiplexing (b) VC assigment

Figure 1: Confined-interference communication on a NoC.

A bufferless NoC is a low power consumption communication fabric. By eliminating VCs (buffers) in routers, bufferless NoCs [9–11] can significantly reduce the power consumption of a NoC. However, as there are no VCs in routers to hold packets, packets have to keep moving on the bufferless NoC. Therefore, when contention occurs between packets on the same output port, some of the packets must be deflected to other output ports, which makes the interference between packets severe. As a consequence, the conventional bufferless NoCs do not support confined-interference communication. Furthermore, as VCs are eliminated, the bufferless NoCs cannot easily accommodate the transfer of multiple class packets in the cache protocol, because they cannot isolate/confine the interference between difference class packets.

Therefore, in this paper, we address the problem of how to achieve confined-interference communication on a bufferless NoC. In order to take advantage of the low power consumption of a bufferless NoC and to achieve confined-interference communication, we propose a novel routing approach called Surfing on a Bufferless NoC (Surf-Bless). The specific novel contributions of this paper are the following:

- We propose a specific assignment and scheduling of domains onto input/output ports, crossbars, and links. This specific assignment and scheduling can be visualized as multiple "waves" (see Figure 3) which move in space and time over

*Corresponding authors

DAC '19, June 2–6, 2019, Las Vegas, NV, USA

ACM ISBN 978-1-4503-6725-7/19/06…$15.00
https://doi.org/10.1145/3316781.3317917

the NoC in a specially designed repetitive pattern. Different domains are assigned to different waves and the packets of a domain "surf" on their own waves, thereby achieving packet routing with no interference between packets belonging to different domains. Our specific repetitive movement of the waves guarantees that packets surfing on any wave do not need to stop and wait at any router along their path to the destination. Thus, VCs are not needed in routers, thereby enabling the utilization of bufferless NoCs to transmit packets.

- We propose an extension for the architecture of a conventional bufferless NoC router that implements in hardware our specific assignment and scheduling introduced in the aforementioned contribution. By adding three simple hardware schedulers in each router, our Surf-Bless routing is implemented in a distributed way, which makes our approach scalable and applicable for large size NoCs.
- By experiments, we show that Our Surf-Bless routing is effective in supporting confined-interference communication and consumes much less energy than Surf-routing [2]. Furthermore, our Surf-Bless overcomes the drawback of the conventional bufferless NoC [9] that can not support the transfer of multiple class packets for the cache coherence protocol.

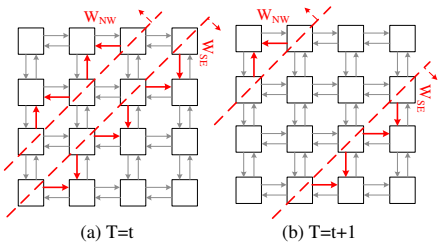
The remainder of the paper is organized as follows: Section 2 gives some background information about the Surf-routing [2] and the BLESS-routing [9] that have inspired our Surf-Bless routing. Section 3 provides an overview of the related work. Section 4 elaborates our novel Surf-Bless routing. Section 5 introduces the experimental setup and presents experimental results. Section 6 concludes this paper.

2 BACKGROUND

In order to better understand the novel contributions of this paper, in this section, we give some background information about the Surf-routing [2] that realizes confined-interference communication which requires VCs (buffers) in routers, and BLESS-routing [9] that is a typical bufferless routing with no support for confined-interference communication.

2.1 Surf-routing

As introduced in Section 1, packets in VCs assigned to a domain can be transferred only in the corresponding time slots in order to achieve isolation in time. As a consequence, packets have to be blocked extra clock cycles at each router to wait for their own time slots, which results in high packet latency. In order to avoid this extra blocking time, Surf-routing [2] assigns and schedules the domains on two kinds of "waves": the north-west wave W_{NW} and the south-east wave W_{SE}. The wave pattern is shown in Figure 2. These waves move in space and in time over the network. The red links (bolded arrows) on the waves indicate that these links are assigned to domain D_0 and only packets in domain D_0 can go through these red links. The packets of domain D_0 "surf" on the waves to the next routers. When packets arrive at the next routers, the waves also arrive at the same routers. Thus, packets can be continuously transferred and do not need to spend too much extra time on waiting for their time slots. Therefore, packet latency in a domain can be reduced. However, VCs in routers are still necessary for the Surf-routing to separately hold packets because when contention occurs between packets in the same domain, some packets must be stored in VCs assigned to the same domain. As a consequence, each domain in Surf-routing needs at least one VC.



(a) T=t (b) T=t+1

Figure 2: The wave pattern in Surf-routing.

2.2 BLESS-routing

By eliminating the VCs (buffers) in a router, BLESS-routing [9] can significantly reduce the power consumption. As there are no VCs to temporarily hold packets, packets in BLESS-routing have to keep moving in the links (or in the router pipeline). When multiple packets contend for the same output port in a router, some of the packets have to be deflected to other output ports, which requires that each packet should always find a free output port to go. This packet deflection is guaranteed because the number of the input ports in a router is equal to the number of the output ports. The packet deflection may cause the livelock problem, i.e., packets keep moving in the network but never reach their destination. In order to avoid the livelock problem, the old-first arbitration policy [12] is used to prioritize packets when they contend for the same output port. When a packet becomes the oldest, it cannot be deflected to other output ports any more. Thus, the livelock is avoided.

3 RELATED WORK

As VCs consume large proportion of the NoC power, several bufferless NoCs [9–11] have been proposed to reduce the power consumption. Based on packet deflection, BLESS [9] eliminates the need for VCs in routers and proves that the bufferless NoC is effective when the links utilization is low. CHIPPER [10] proposes a permutation network deflection routing which simplifies the router structure and reduces the bandwidth overhead caused by the old-first arbitration policy [12]. Without deflecting packets but dropping packets, Runahead [11] further simplifies the router structure and it can be used to achieve low latency communication. By eliminating the VCs, [9–11] can significantly reduce the NoC power consumption. However, as there are no VCs to temporarily hold packets, [9–11] cannot support confined-interference communication, thus cannot be used in composable many-core systems. In contrast, our Surf-Bless routing is an approach to achieve confined-interference communication on a bufferless NoC. Thanks to our Surf-Bless routing, it becomes possible for a bufferless NoC to be used in composable systems to achieve energy-efficient communication. Thus, compared with [9–11], our Surf-Bless routing extends the applicability of the bufferless NoC.

As introduced in Section 2.1, by scheduling packets in different domains onto different waves, the Surf-routing [2] can reduce the packet latency caused by waiting for the corresponding time slots. As an extension of Surf-routing, the router pipeline in PhaseNoC [3] is time-multiplexed by all domains. PhaseNoC reduces the part of the hardware overhead caused by separating the NoC resources for different domains. However, each domain in Surf-routing and PhaseNoC needs at least one VC, which may lead to a large number of buffers and cause high power consumption in case of multiple domains. In contrast, based on the bufferless NoC, our Surf-Bless routing achieves confined-interference communication without the need of VCs. Thus, our Surf-Bless routing is much more power efficient than the approaches in [2] and [3].

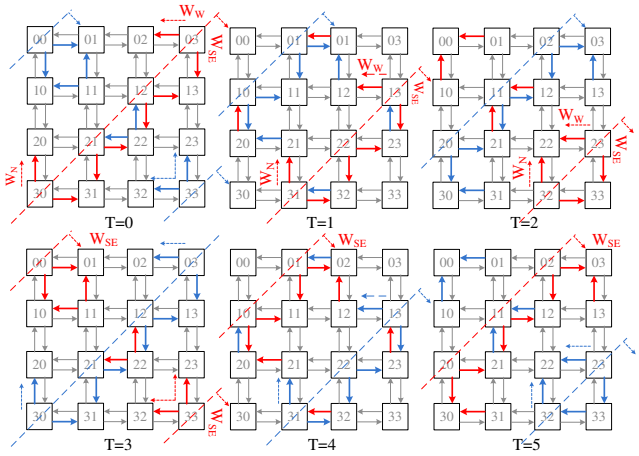


Figure 3: Wave pattern in Surf-Bless routing.

By reserving time slots along a routing path, [13–15] realize contention-free routing and guarantee that there is no interference between packets. So, the NoC provides composable and predictable services. As data flows are time-multiplexed to use the NoC resources, the router structures in [13–15] are simple and do not need too much buffers, so the power consumption is low. However, to achieve the contention-free routing, [13–15] need to globally schedule all data flows on the NoC at design time. As a consequence, the contention-free routing in [13–15] is only applicable to static data (packets) traffic, where traffic information, such as traffic patterns, the data volume of each data flow, etc., are known at design time. So, for dynamic traffic patterns, i.e., traffic unknown at design time, [13–15] are not applicable. In contrast, our Surf-Bless routing schedules domains on the NoC and does not need to schedule each packet. Thus, the scheduling in our Surf-routing is simpler than [13–15]. Furthermore, our Surf-Bless routing does not need the aforementioned traffic information at design time to schedule domains onto a NoC. Thus, our Surf-Bless routing is applicable to dynamic traffic patterns as well, providing composability services to a system but cannot provide complete predictability for dynamic traffic.

4 SURF-BLESS ROUTING

The key idea of our novel Surf-Bless routing is to assign and schedule domains onto waves that move in space and time over the NoC in a specially designed repetitive pattern. Packets of a domain can go only through the input ports, crossbars, output ports, and links on the waves assigned to the same domain. Thus, as there is no interference between waves, it is guaranteed that there is no interference between different domains. Furthermore, based on the special wave pattern in our Surf-Bless routing, it is guaranteed that in a router, the number of input ports assigned to one domain at the current clock cycle, is equal to the number of output ports assigned to the same domain at the next clock cycle. This makes it possible to always deflect packets if needed in order to keep packets moving in the network instead of holding them temporarily in VCs. Thus, we achieve confined-interference communication on a bufferless NoC, i.e., a NoC without VCs.

4.1 Wave pattern in Surf-Bless

In order to keep packets of a domain traveling to their destinations without the need to stop and wait at routers, we assign and schedule domains onto NoC resources in a special repetitive pattern, which can be visualized as the “waves" shown in Figure 3. For the sake of clarity, in Figure 3, we show only two waves which are distinguished by the blue color and the red color. These waves move in space and time over the network and the pattern repeats after 6 time slots T. In the following explanation, we take the red wave as an example to describe our special wave pattern. The wave consists of three sub-waves, the south-east sub-wave (W_{SE}), the north sub-wave (W_N), and the west sub-wave (W_W). These sub-waves of a wave must respect the following two rules: **Rule-1**: when W_{SE} reaches the routers on the south-border of the network or the routers on the east-border of the network, W_N at the router on the south-border and W_W at the router on the east-border are triggered. For example, at time slot $T = 0$, W_{SE} reaches router 30 on the south-border and router 03 on the east-border. At the same time, W_N at router 30 and W_W at router 03 are triggered to move over the network. **Rule-2**: when W_N reaches a router on the north-border or W_W reaches a router on the west-border, W_{SE} must arrive at the corresponding routers as well. For example, at time slot $T = 4$, W_N and W_W reach the north-border at router 01 and the west-border at router 10, respectively. At the same time, W_{SE} reaches the same router 01 and 10. Constrained by these two rules, a wave moves over the network repetitively, like a reverberating wave. As there is no any overlapping between any two waves, it can be guaranteed that there is no interference between domains that are assigned to different waves.

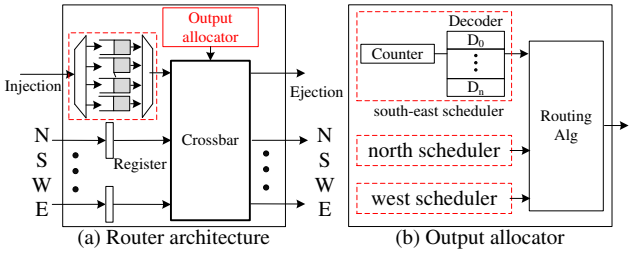
In contrast to the wave pattern in the Surf-routing shown in Figure 2, the wave pattern in our Surf-Bless routing guarantees that a router has a certain number of input ports receiving packets from the links belonging to a given wave at time slot (clock cycle) T. At the next time slot $T + 1$, this router has the same number of output ports sending packets to the links belong to the same wave. Thus, thanks to our special wave pattern, it is possible to deflect packets when contention occurs on the same output port. Therefore, the packets can keep moving in the network and we can achieve confined-interference communication on a bufferless NoC.

4.2 Router Architecture

In order to assign and schedule domains on the waves, shown in Figure 3, to support confined-interference communication, we extend a conventional bufferless router, as shown in Figure 4. Our extensions are indicated by the red color. The router consists of five input ports, five output ports, one output allocator and one crossbar.

In the input ports at the directions of north (N), south (S), west (W), and east (E), VCs are eliminated but one register per input port is used to construct the router pipeline. For the injection input port, there are multiple VCs. These VCs are necessary for a bufferless NoC to temporarily hold packets coming from the network interface. These VCs can be put in the router [9] or put in the network interface [10]. We put these VCs in the router to simplify the control of the packet injection. In contrast to [9] and [10] with only one VC in each injection input port, we utilize multiple VCs and each VC is assigned to one domain. In this way, the packets in a domain cannot be blocked by packets in other domains, i.e., the Head-of-Line blocking [12] between packets in different domains is avoided.

The output allocator is used to implement our novel wave pattern introduced in Section 4.1. Based on our routing algorithm described in Section 4.3, it allocates the output ports for each incoming packet. In order to implement our wave pattern, three schedulers: the south-east scheduler, the north scheduler, and the west scheduler, are realized in the output allocator as shown in Figure 4(b). Each scheduler corresponds to one sub-wave and is responsible for a pair of input ports and output ports. The south-east scheduler corresponds to sub-wave W_{SE} and is responsible for



(a) Router architecture (b) Output allocator

Figure 4: Router architecture in Surf-Bless.

the pair $\{input = \{N, W, Injection\}, output = \{S, E, Ejection\}\}$. This implies that packets can be injected or ejected only on the W_{SE} sub-wave. The north scheduler and the west scheduler correspond to W_N and W_W and are responsible for $\{input = \{S\}, output = \{N\}\}$ and $\{input = \{E\}, output = \{W\}\}$, respectively.

These schedulers have similar hardware structure. Taking the south-east scheduler as an example, shown in Figure 4(b), it consists of a counter and a decoder. The counter cyclically counts from 0 to $S_{max} - 1$. S_{max} is the maximal number of waves, which is determined by the NoC size, the pipeline stage of router, and link delay. It can be calculated by $S_{max} = 2 \times P \times (N - 1)$, where P is the hop delay in clock cycles (the delay of a packet to go through one router and one link), and N is the number of routers in one dimension on a $N \times N$ NoC. By setting specific initial values for the counters in the router, we can realize the wave pattern shown in Figure 3. The initial value for each counter in the router (x, y) (x and y are the router position in each dimension on the NoC) is computed by the following equations:

$$Initial_{SE} = (S_{max} \times P - P(x + y)) \mod S_{max} \quad (1)$$

$$Initial_{W} = (S_{max} \times P + P(x - y)) \mod S_{max} \quad (2)$$

$$Initial_{N} = (S_{max} \times P - P(x - y)) \mod S_{max} \quad (3)$$

The decoder is a table and is used to assign different waves to different domains. Based on these schedulers in each router, we implement our special assignment and scheduling in a distributed way without the need of global control.

4.3 Surf-Bless routing algorithm

The Routing Algorithm unit in the output allocator in Figure 4(b) is used to allocate output ports for incoming packets. In our Surf-Bless approach, a packet in an input port assigned to a domain can go only through the output ports that are assigned to the same domain. To respect this rule, we propose the following Surf-Bless routing algorithm, which consists of two steps:

Step-1: select the highest priority packet from input ports; For simplicity, our Surf-Bless routing uses the old-first arbitration policy [12]. Packets carry "age" information, i.e., the longer the packet moves in the network the older it becomes. The oldest packet has the highest priority. The packets in the injection input ports have the lowest priority.

Step-2: choose an output port in the same domain; First, our Surf-Bless routing algorithm uses X-Y routing to determine the output port for the packet selected in Step-1. If the input port and the determined output port are in the same domain checked by comparing the outputs of the schedulers as well as the output port is not granted to another packet, the output port is granted to the selected packet. If this check fails, our Surf-Bless routing tries the Y-X routing. If the same check fails again, one of the free output ports assigned to the same domains is randomly granted to the packet, thereby deflecting the packet. For a packet in the injection input port, if there is one free output port that is assigned to the same domain with the injection input port, the packet in the corresponding VC can be transferred, otherwise the packet is blocked in the corresponding VC.

As there are no VCs (buffers) for the N, S, W, and E input ports in our routers, the routers need to deflect packets, which may cause livelock and deadlock problems. In Step-1, our Surf-Bless uses the old-first arbitration policy [12] which guarantees that our Surf-Bless routing is deadlock and livelock free. Dependent packets may cause the protocol deadlock in a bufferless NoC. For example, the protocol packets described in [16] consist of a request packet and a reply packet, where the reply packet must be injected to the network before a new request packet arrives, otherwise the new request packet cannot be ejected to the network interface. However, in a bufferless NoC, a new request packet in the NoC has higher priority than the reply packets staying at the network interface, so a new request packet blocks the reply packets to be injected into the NoC. As a consequence, cyclic dependency occurs between the request packet and the reply packet, which may result in deadlock. In our Surf-Bless routing, such dependent packets can be separately assigned to different domains. As there is no interference between packets in different domains, the cyclic dependency between packets is removed and the deadlock is avoided.

Table 1: Parameters.

Network topology	8×8 mesh
Router	2-stage and 4-stage pipelines
Virtual channel	1 ctrl VC and 2 data VCs
Input buffer size	1-flit/ ctrl VC, 5-flit / data VC
Routing algorithm	X-Y DOR, Surf and Surf-Bless
Link bandwidth	128 bits/cycle
Private I/D L1$	32 KB
Shared L2 per bank	256 KB
Cache block size	16 Bytes
Coherence protocol	Two-level MESI
Memory controllers	4, located one at each corner

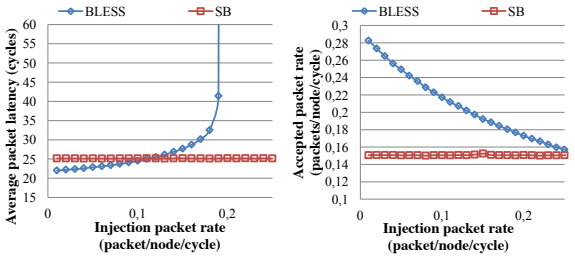
5 EXPERIMENTAL RESULTS

In order to evaluate our approach in terms of performance and energy consumption, we have implemented our Surf-Bless routing on the full system simulator GEM5 [17]. The NoC model and power model is based on Garnet2.0 [18] and Dsent [19], respectively. The key parameters used in our experiments are shown in Table 1. Considering the previous works [9] and [10], the router pipeline delay in the bufferless NoC is set to 2 clock cycles. For a conventional virtual channel router, we choose a 4-stage pipeline router.

For comparison purpose, we have implemented the following approaches: (1) WH [12]: the baseline wormhole NoC, which does not support confined-interface communication; (2) Surf [2]: a confined-interference NoC with the Surf routing briefly introduced in Section 2.1; (3) BLESS [9]: the bufferless baseline NoC briefly introduced in Section 2.2, which does not support confined-interference communication; (4) SB: our Surf-Bless approach supporting confined-interference communication on a bufferless NoC.

5.1 Evaluation on Synthetic Workloads

In this section, we evaluate our approach in terms of the ability of supporting confined-interference communication, energy consumption, average packet latency, and throughput. Based on our NoC configuration in Table 1 and the S_{max} equation introduced in Section 4.2, there are $S_{max} = 2 \times 3 \times (8 - 1) = 42$ waves. The domains are equally and evenly assigned to these waves. For example, in section 5.1.1 there are two domains. One domain is assigned to the even waves and the other one is assigned to the odd waves. The uniform random traffic pattern described in [12] is used to inject 1-flit packets into the NoCs in our experiments.



(a) Latency on the other domain. (b) Throughput on the other domain.

Figure 5: Non-interference between domains.

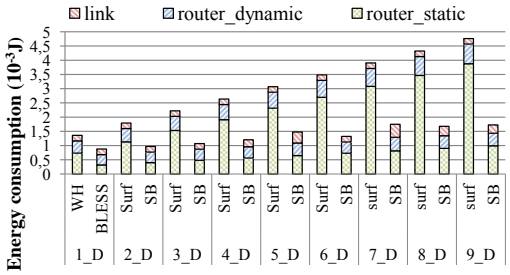


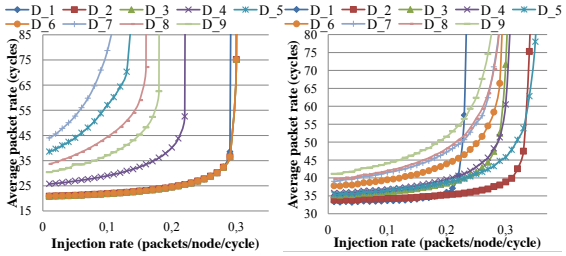
Figure 6: Energy consumption.

5.1.1 Confined-interference Communication. In order to test the ability of confining the interference between packets from different domains in our SB, we use two domains. One domain is regarded as interference traffic. By observing the latency and throughput of the other domain, we can check if there is any interference between packets in the different domains.

Figure 5(a) and Figure 5(b) show the average packet latency and the maximal throughput provided by SB and BLESS for the other domain considering different injection rates of the interference traffic, respectively. By increasing the injection rate of the interference traffic, the average packet latency and the throughput provided by our SB for the other domain stays constant, which indicates that SB is effective on isolating the interference between domains. In contrast, as BLESS does not support confined-interference communication, the average packet latency and throughput provided by BLESS for the other domain are significantly impacted by the interference traffic.

5.1.2 Energy Consumption under Different Domains. In this section, we evaluate the energy consumption considering one domain (D_1), $\cdots$, nine domains (D_9). Each domain has one VC with the size of 4-flit. The domains are equally and evenly assigned to the waves. The packets are equally assigned/injected to each domain. In our experiment, we evaluate the energy consumption of the NoCs in a time period of 1 million clock cycles (the frequency is $1GHz$) under packet injection rate 0.05 packets/node/cycle.

Figure 6 shows the energy consumption across D_1, D_2, $\cdots$, D_9. The NoC energy consumption is broken down into dynamic and static router energy consumption and total link energy consumption. Compared to Surf with different domains, the total energy consumption in SB is significantly reduced. This energy reduction increases with the number of domains increasing. This is because, by eliminating the VCs for the N, S, W, and E input ports in a router, the router in our SB has much lower static energy consumption and simpler architecture than the router in Surf. Furthermore, with the number of domains increasing, in our SB, only injection input ports need to increase the number of VCs to separately hold packets for different domains. While in Surf, all of the input ports need to increase VCs to separately hold packets for different domains, which causes sharply increase of the static energy consumption. On the other hand, compared with BLESS, as SB needs multiple VCs in the injection, Surf-bless causes higher static energy consumption than BLESS.



(a) BLESS (D_1) and Surf-Bless (b) WH (D_1) and Surf

Figure 7: Latency across different number of domains.

5.1.3 Average Packet Latency under Different Domains. Figure 7(a) and Figure 7(b) shows the average packet latency of Surf-Bless and Surf under 1_D, D_2, $\cdots$, D_9, respectively. In Figure 7(a), the curves of D_2, D_3, and D_6 are overlapped and have the highest throughput. D_4, D_5, D_7, D_8, and D_9 have higher average packet latency and lower throughput. This is because, in SB, the packet injection/ejection happens only on the south-east sub-wave W_{SE}. This means that when a packet in a domain reaches its destination router on the north sub-wave W_N or the west sub-wave W_W and W_{SE} assigned to the same domain has not reached this router yet, this packet cannot be ejected and has to be deflected to a downstream router, in spite of the fact that this packet has already reached its destination router. This negative impact does not happen for D_2, D_3, and D_6, because sub-wave W_W and W_{SE} always reach a router at the same time with sub-wave W_{SE}. However, for the D_4, D_5, D_7, d_8, and D_9, this negative impact is serious and causes high packet latency increase.

Compared Figure 7(a) with Figure 7(b), when the number of domains is 5,7,8, and 9, SB has higher performance penalty than Surf in terms of the average packet latency and the throughput. This is because of the shortage of our SB to assign domains on waves. As a consequence, our SB is effective in supporting confined-interference communication and has comparable performance with Surf when the number of domains is 2,3,4, and 6.

5.2 Transfer of Multiple Class Packets

In the cache coherence protocol, there are multiple class packets, which should be separately hold in different VCs and transferred in different virtual network [12]. It is necessary to guarantee there is no protocol deadlock. However, as there are no VCs, the conventional bufferless NoCs does not support the transfer of multiple class packets. While, by supporting confined-interference communication, our SB can easily support the transfer of different class packets for the cache coherence protocol.

To do so, we apply our SB on transferring different class packets for the MESI protocol [20], which needs two data virtual networks to transfer 5-flit packets and one control virtual network to transfer 1-flit packets. To separate packets of different virtual networks, we set three domains and packets of each virtual network are assigned/injected to one separated domain. There are 42 waves in our experiment. To continuously transfer 5-flit packets, packets of one data virtual network are assigned to three sets of waves $\{0,1,2,3,4\}$, $\{15,16,17,18,19\}$, and $\{30,31,32,33,34\}$, and packets of the other data virtual network are assigned to another three sets of waves $\{7,8,9,10,11\}$, $\{22,23,24,25,26\}$, and $\{37,38,39,40,41\}$. It should be noted that the 5-flit packets only choose the output port assigned at the begin of the wave sets. The 1-flit packets of the control virtual

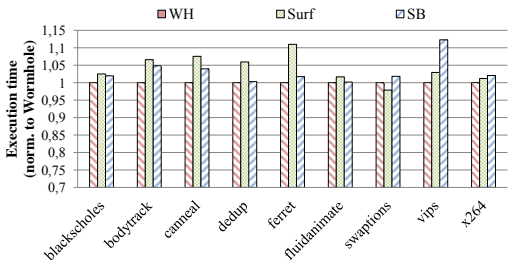


Figure 8: Application execution time.

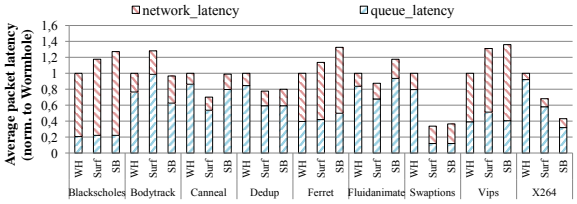


Figure 9: Latency.

network are assigned to the rest of the waves. As BLESS does not support the transfer of multiple class packets with various number of flits, BLESS is not considered in this experiment.

5.2.1 Application Execution Time. Figure 8 shows the execution time of the nine applications in Parsec [21], which is normalized to WH (each input port has two data VCs and one control VC to build three virtual networks) as the baseline. Compared with WH, our SB causes an average of 3.23% performance penalty, which is less than the 4.13% performance penalty in Surf. In two applications (**dedup** and **fluidanimate**), our SB achieves less than 1% performance penalty. In **vips**, our SB has its largest performance penalty of 12.27%. While, for Surf, the largest performance penalty of 10.99% happens for **ferret**. An interesting observation is that Surf achieves 2.14% performance improvement for **swaption**. This improvement can be attributed to the acceleration of some packets'transmission.

5.2.2 NoC Packet Latency. Figure 9 shows the average packet latency for the nine applications. The average packet latency is broken down into the blocking time in the network interface (queue latency) and the transmission time on the NoC (network latency). Compared with WH, the average packet latency in Surf and SB is significantly reduced in **dedup**, **swaption**, and **x264**, while, in **blackscholes**, **ferret**, and **vips**, the average packet latency increases. Most of the latency decrease comes from the queue latency reduction of the packets, while the latency increase is caused by the increase of the network latency. This is because based on the wave assignment in Surf and SB, the network resources are reserved by different virtual networks, which guarantees the packet transmission in each virtual network. However, these resource reservation mechanisms also undermine the efficiency of the networks and cause the network latency increasing.

By analyzing the result reports of GEM5, we found that most of the latency decrease comes from the queue latency reduction of the control (1-flit) packets, which may not be critical to improve the application performance. As a consequence, even though the average packet latency in Surf and SB is reduced for some applications, Surf and SB still cause performance penalty in the application execution time.

5.2.3 NoC Energy Consumption. Figure 10 shows the NoC energy consumption for the nine applications. It is broken down into three parts, the static and dynamic power consumption of routers and the total power consumption of links. As shown in Figure 10, SB consumes much less energy than the related approaches. Compared

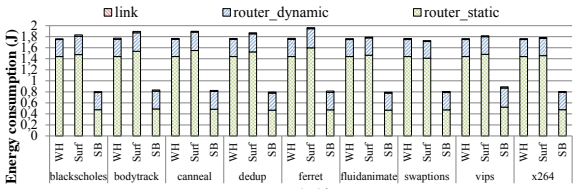


Figure 10: Energy consumption.

with WH, our SB reduces total energy consumption with 53.6% on average. This high energy reduction is achieved by eliminating the VCs for the N, S, W, and E input ports in a route to reduce the static energy consumption. In contrast, Surf has higher energy consumption than WH. The link energy consumption is negligible, because we use the low workload mode in Parsec.

6 CONCLUSION

In this paper, we propose the Surf-Bless routing and extend the router architecture to implement Surf-Bless in a distributed way. Based on our Surf-Bless approach, a conventional bufferless NoC is extended to support confined-interference communication. Furthermore, by taking advantage of the low power consumption of a bufferless NoC, our Surf-Bless routing achieves confined-interference communication with low energy consumption.

7 ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 61672526, Grant 61572508, in part by the Research Project of NUDT under Grant ZK17-03-06, and in part by the Science and Technology Innovation Project of Hunan Province under Grant 2018RS3083.

REFERENCES

[1] A. Hansson *et al.*, "Compsoc: A template for composable and predictable multi-processor system on chips," *TODAES*, 2009.
[2] H. M. Wassel *et al.*, "Surfnoc: a low latency and provably non-interfering approach to secure networks-on-chip," in *ACM SIGARCH*, 2013.
[3] A. Psarras *et al.*, "Phasenoc: Versatile network traffic isolation through tdm-scheduled virtual channels," *IEEE TCAD*, 2016.
[4] L. Chen *et al.*, "Power punch: Towards non-blocking power-gating of noc routers," in *HPCA*, IEEE, 2015.
[5] P. Wang *et al.*, "A novel approach to reduce packet latency increase caused by power gating in network-on-chip," in *NOCS*, 2017.
[6] Y. Hoskote *et al.*, "A 5-ghz mesh interconnect for a teraflops processor," *IEEE Micro*, 2007.
[7] B. K. Daya *et al.*, "Scorpio: a 36-core research chip demonstrating snoopy coherence on a scalable mesh noc with in-network ordering," in *ISCA*, 2014.
[8] H. Esmaeilzadeh *et al.*, "Dark silicon and the end of multicore scaling," in *ACM SIGARCH Computer Architecture News*, 2011.
[9] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *ACM SIGARCH Computer Architecture News*, ACM, 2009.
[10] C. Fallin *et al.*, "Chipper: A low-complexity bufferless deflection router," in *HPCA*, IEEE, 2011.
[11] Z. Li *et al.*, "The runahead network-on-chip," in *HPCA*, IEEE, 2016.
[12] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.
[13] K. Goossens *et al.*, "Æthereal network on chip: concepts, architectures, and implementations," *IEEE Design & Test of Computers*, 2005.
[14] A. Hansson *et al.*, "aelite: A flit-synchronous network on chip with composable and predictable services," in *DATE*, 2009.
[15] R. A. Stefan *et al.*, "daelite: A tdm noc supporting qos, multicast, and fast connection set-up," *IEEE Transactions on Computers*, 2014.
[16] A. Hansson *et al.*, "Avoiding message-dependent deadlock in network-based systems on chip," *VLSI design*, 2007.
[17] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, 2011.
[18] N. Agarwal *et al.*, "Garnet: A detailed on-chip network model inside a full-system simulator," in *ISPASS*, 2009.
[19] C. Sun *et al.*, "Dsent-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *NOCS*, 2012.
[20] M. S. Papamarcos *et al.*, "A low-overhead coherence solution for multiprocessors with private cache memories," in *ACM SIGARCH Computer Architecture News*, ACM, 1984.
[21] C. Bienia *et al.*, "The parsec benchmark suite: Characterization and architectural implications," in *PACT*, ACM, 2008.