Chapter 6

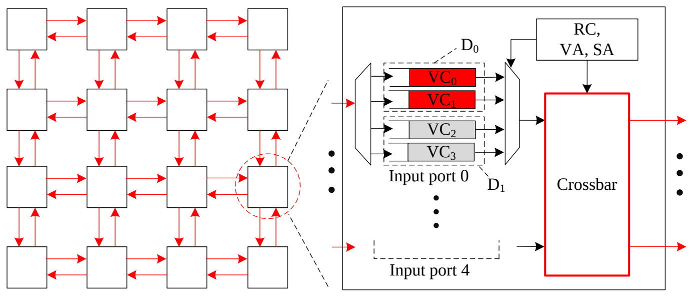
Energy-Efficient Confined-interference Communication

Peng Wang, Sobhan Niknam, Sheng Ma, Zhiying Wang, Todor Stefanov,
"Surf-Bless: A Confined-interference Routing for Energy-Efficient Communication in NoCs",
in Proceeding of the 56th Annual Design Automation Conference (DAC), Las Vegas, United States, 2019,
Winner of 2019 HiPEAC Paper Award

In this chapter, we present our Surfing on a Bufferless NoC (Surf-Bless) routing approach, which corresponds to **Contribution 4** introduced in Section 1.4 to solve the research **Problem 2** formulated in Section 1.3.2. The remainder of this chapter is organized as follows. Section 6.1 further introduces the high power consumption problem caused by supporting confined-interference communication on conventional NoCs. Section 6.2 summarizes the main contributions in this chapter. To better understand our Surf-Bless routing approach, Section 6.3 gives some background information about the Surf-routing [WGO$^+$13] and the BLESS-routing [MM09], that have inspired our Surf-Bless routing approach. Section 6.4 provides an overview of the related work. It is followed by Section 6.5, which elaborates our novel Surf-Bless routing approach. Section 6.6 introduces the experimental setup and presents experimental results. Finally, Section 6.7 concludes this chapter.

6.1 Problem Statement

A Network-on-Chip (NoC) with low latency, high bandwidth, and good scalability is a promising communication infrastructure for large size many-core systems.



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

Figure 6.1: Confined-interference communication on a NoC.

However, as the NoC resources are shared by different packets, there may be significant interference between packets, which influences the 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 [HGBH09]. 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 [WGO$^+$13, P$^+$16]. 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 resources 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 store 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 6.1, 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 stored 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 6.1(a) and 6.1(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. This leads to a large number of buffers when there are a lot of domains and causes high static and dynamic power consumption [CZPP15, WNWS17]. For example, the NoC with two 16-flit VCs per input port in the Intel 80-tile chip [HVS+07] and the NoC with four 1-flit VCs and two 3-flit VCs per input port in the SCORPIO chip [DCS+14] consume up to 28% and 19% of the total system power, respectively, see Table 1.1. In fact, such high power consumption of a NoC has become the major bottleneck that prevents applying NoCs on many-core systems [EBA+11]. Therefore, it is critical to implement confined-interference communication with low power/energy consumption.

A bufferless NoC is a low power consumption communication fabric. By eliminating VCs (buffers) in routers, bufferless NoCs [MM09, FCM11, LSMJ16] can significantly reduce the power consumption of a NoC. However, as there are no VCs in routers to store 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 a cache coherence protocol, because they cannot isolate/confine the interference between difference class packets.

6.2 Contributions

Considering the discussion above, in this chapter, 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 chapter 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 6.3) that move in space and time over 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 approach 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 approach is effective in supporting confined-interference communication and consumes much less energy than Surf-routing [WGO+13]. Furthermore, our Surf-Bless approach overcomes the drawback of the conventional bufferless NoC [MM09] which cannot support the transfer of multiple class packets for a cache coherence protocol.

6.3 Background

In order to better understand the novel contributions of this chapter, in this section, we give some background information about the Surf-routing [WGO+13] which realizes confined-interference communication and requires VCs (buffers) in routers, as well as we briefly describe the BLESS-routing [MM09] which is a typical bufferless routing approach with no support for confined-interference communication.

6.3.1 Surf-routing

As introduced in the begin of this chapter, 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 [WGO+13] 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 6.2. 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, the packet latency in a domain can be reduced.

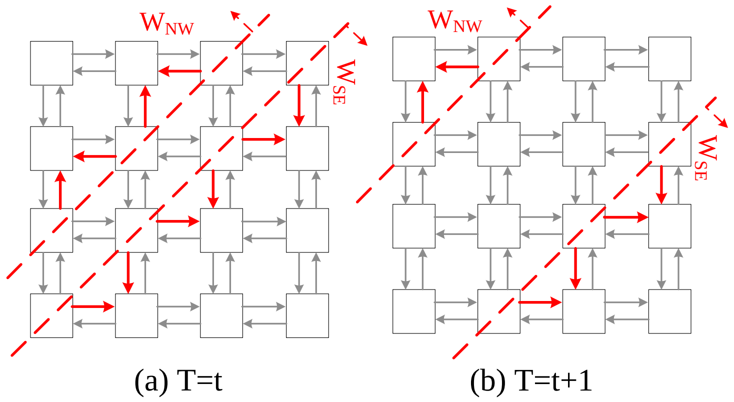


Figure 6.2: The wave pattern in Surf-routing.

However, VCs in routers are still necessary for the Surf-routing to separately store 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.

6.3.2 BLESS-routing

By eliminating the VCs (buffers) in a router, BLESS-routing [MM09] can significantly reduce the power consumption. As there are no VCs to temporarily store 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 a livelock problem, i.e., packets keep moving in the network but never reach their destination. In order to avoid such livelock problem, the old-first arbitration policy [DT04] 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.

6.4 Related Work

As VCs (buffers) consume a large portion of the NoC power, several bufferless NoCs [MM09, FCM11, LSMJ16] have been proposed to reduce the power consumption. Based on packet deflection, BLESS [MM09] eliminates the need for VCs in routers and proves that the bufferless NoC is effective when the links utilization is low. CHIPPER [FCM11] proposes a permutation network deflection routing which simplifies the router structure and reduces the bandwidth overhead caused by the old-first arbitration policy [DT04]. Without deflecting packets but dropping packets, Runahead [LSMJ16] further simplifies the router structure and it can be used to achieve low latency communication. By eliminating the VCs, [MM09, FCM11, LSMJ16] can significantly reduce the NoC power consumption. However, as there are no VCs to temporarily store packets, [MM09, FCM11, LSMJ16] 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 approach, it becomes possible for a bufferless NoC to be used in composable systems to achieve energy-efficient communication. Thus, compared with [MM09, FCM11, LSMJ16], our Surf-Bless routing extends the applicability of the bufferless NoCs.

As introduced in Section 6.3.1, by scheduling packets in different domains onto different waves, the Surf-routing [WGO+13] can reduce the packet latency caused by waiting for the corresponding time slots. As an extension of Surf-routing, the router pipeline in PhaseNoC [P+16] 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 a bufferless NoC, our Surf-Bless routing approach achieves confined-interference communication without the need of VCs. Thus, our Surf-Bless routing approach is much more power efficient than the approaches in [WGO+13] and [P+16].

By reserving time slots along a routing path, [GDR05, HSG09, SMG14] 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 [GDR05, HSG09, SMG14] are simple and do not need too much buffers, so the power consumption is low. However, to achieve the contention-free routing, [GDR05, HSG09, SMG14] need to globally schedule all data flows on the NoC at design time. As a consequence, the contention-free routing in [GDR05, HSG09, SMG14] 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, [GDR05, HSG09, SMG14] are not applicable. In contrast, our Surf-Bless routing approach schedules domains on the NoC and does not need to schedule each packet. Thus, the scheduling in our Surf-routing approach is simpler than [GDR05, HSG09, SMG14]. Furthermore, our Surf-Bless routing approach 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.

6.5 Surf-Bless Routing Approach

The key idea of our novel Surf-Bless routing approach 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 approach, 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 storing them temporarily in VCs. Thus, we achieve confined-interference communication on a bufferless NoC, i.e., a NoC without VCs.

6.5.1 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 6.3. For the sake of clarity, in Figure 6.3, 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

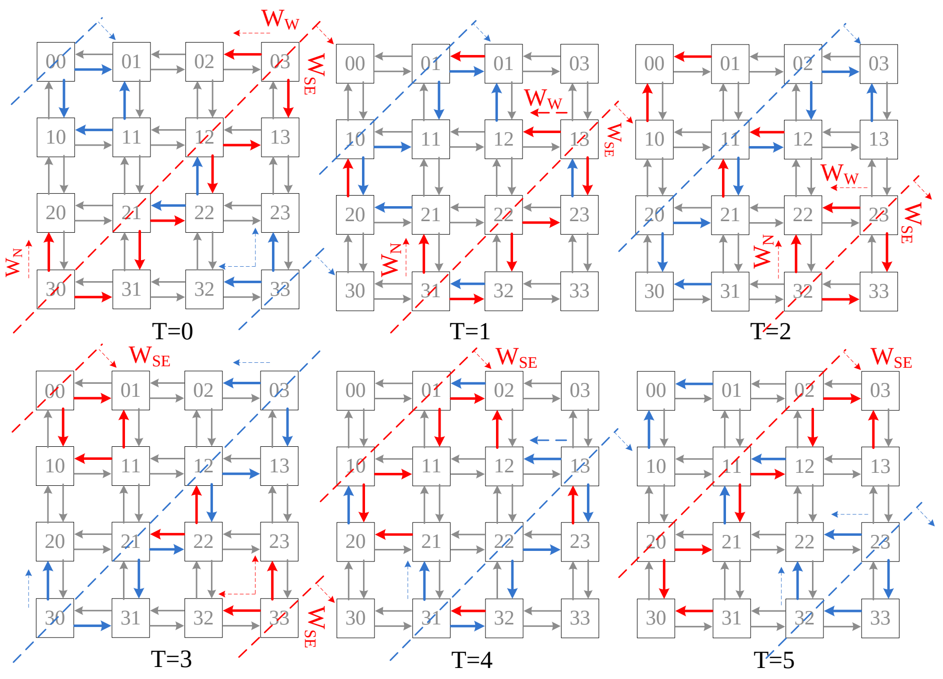


Figure 6.3: Wave pattern in Surf-Bless routing.

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 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 6.2, 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.

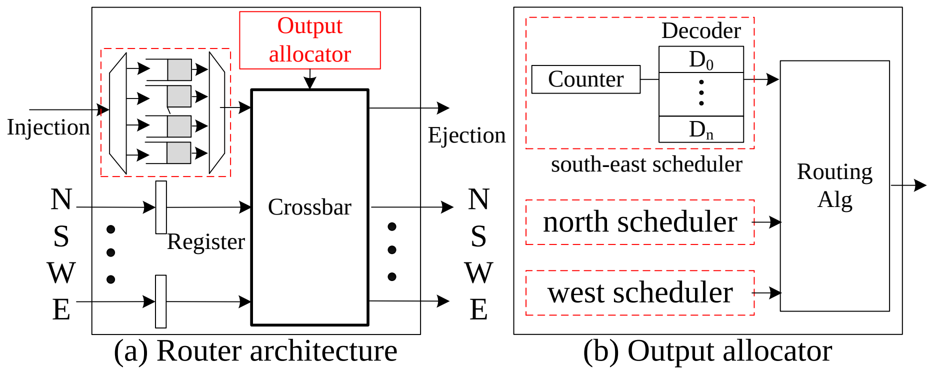


Figure 6.4: Router architecture in Surf-Bless.

6.5.2 Router Architecture

In order to assign and schedule domains on the waves, shown in Figure 6.3, to support confined-interference communication, we extend a conventional bufferless router, as shown in Figure 6.4. 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 store packets coming from the network interface. These VCs can be put in the router [MM09] or put in the network interface [FCM11]. We put these VCs in the router to simplify the control of the packet injection. In contrast to [MM09] and [FCM11] 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 [DT04] between packets in different domains is avoided.

The output allocator is used to implement our novel wave pattern introduced in Section 6.5.1. Based on our routing algorithm described in Section 6.5.3, it allocates the output ports for each incoming packet. In order to implement our wave pattern, three schedulers, i.e., the south-east scheduler, the north scheduler, and the west scheduler, are realized in the output allocator as shown in Figure 6.4(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 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 6.4(b), it consists of a counter and a decoder. The counter cyclically counts from 0 to $S_{max} - 1$ indicating the wave index of the current wave at the router. S_{max} is the maximal number of waves, which is determined by the NoC size, the pipeline stages of a router, and the 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. (The maximal number of waves is also equal to the repeat period of a wave. For example, in Figure 6.3, the repeat period of the red wave is 6 time slots (the hop delay $P = 1$) in this example, so the maximal number of waves in this example is 6.) By setting specific initial values for the counters in the router, we can realize the wave pattern shown in Figure 6.3. The initial value for each counter in a router (x, y) (x and y are the router positions in each dimension on the NoC) is computed by the following equations:

$$Initial_{SE} = (S_{max} - P \times (x + y)) \mod S_{max} \tag{6.1}$$

$$Initial_{W} = (S_{max} + P \times (x - y)) \mod S_{max} \tag{6.2}$$

$$Initial_{N} = (S_{max} - P \times (x - y)) \mod S_{max} \tag{6.3}$$

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.

6.5.3 Surf-Bless routing algorithm

The Routing Algorithm unit in the output allocator in Figure 6.4(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 [DT04]. 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 algorithm uses the old-first arbitration policy [DT04] which guarantees that our Surf-Bless routing approach is deadlock and livelock free. Dependent packets may cause the protocol deadlock in a bufferless NoC. For example, the protocol packets described in [HGR07] 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 approach, 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.

6.6 Experimental Results

In order to evaluate our approach in terms of the performance and the energy consumption, we have implemented our Surf-Bless routing in the full-system simulator GEM5 [BBB$^+$11]. The NoC model and power model are based on Garnet2.0 [AKPJ09] and Dsent [SCK$^+$12], respectively. The key parameters used in our experiments are shown in Table 6.1. Considering the previous works [MM09] and [FCM11], 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 [DT04]: the baseline wormhole NoC, which does not support confined-interface

Table 6.1: 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
Flit size	128 bits
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

communication; (2) SURF [WGO$^+$13]: a confined-interference NoC with the Surf routing briefly introduced in Section 6.3.1; (3) BLESS [MM09]: the bufferless baseline NoC briefly introduced in Section 6.3.2, which does not support confined-interference communication; (4) SB: our Surf-Bless approach supporting confined-interference communication on a bufferless NoC.

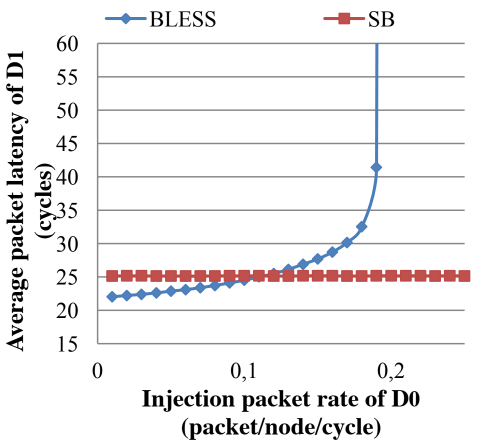
6.6.1 Evaluation on Synthetic Workloads

In this section, we evaluate our approach in terms of energy consumption, average packet latency, and the ability of supporting confined-interference communication. Based on our NoC configuration in Table 6.1 and the S_{max} equation introduced in Section 6.5.2, there are $S_{max} = 2 \times 3 \times (8-1) = 42$ waves. The domains are equally and evenly assigned to these waves. The uniform random traffic pattern described in [DT04] is used to inject 1-flit packets into the NoCs in our experiments.

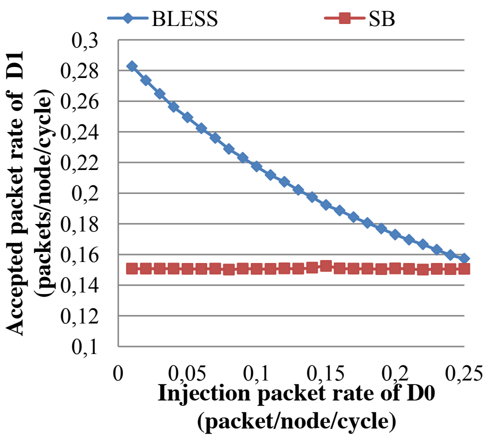
Confined-interference Communication

In order to test the ability of confining the interference between packets from different domains in our SB, we use two domains D_0 and D_1. Domain D_0 is assigned to the even waves and domain D_1 is assigned to the odd waves. Domain D_0 is regarded as interference traffic. By observing the latency and throughput of domain D_1, we can check if there is any interference between packets in the different domains D_0 and D_1.

Figure 6.5(a) and Figure 6.5(b) show the average packet latency and the maximal throughput provided by our SB and BLESS for domain D_1 considering different injection rates of the traffic generated by domain D_0, respectively. By increasing the injection rate of domain D_0, the average packet latency and the throughput provided by our SB for domain D_1 stays constant, which indicates that SB is effective and



(a) Latency of domain D_1.



(b) Throughput of domain D_1.

Figure 6.5: Non-interference between domains.

guarantees no interference between domains. In contrast, as BLESS does not support confined-interference communication, the average packet latency and throughput provided by BLESS for domain D_1 are significantly impacted by the traffic of domain D_0.

Energy Consumption under Different Number of Domains

In this section, we evaluate the energy consumption considering different number of domains, from one domain (1_D) up to nine domains (9_D). In SURF, each input port (N, S, W, E, and Injection) in a router has multiple 4-flit VCs where VC corresponds to one domain. In contrast, in our SB, as shown in Figure 6.4, only the Injection input port has multiple 4-flit VCs where corresponds to one domain. 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 of 0.05 packets/node/cycle.

Figure 6.6 shows the energy consumption across 1_D, 2_D, $\cdots$, 9_D. The NoC energy consumption is broken down into the dynamic and the static router energy consumption and the total link energy consumption. Compared to SURF with different domains, the NoC energy consumption in our SB is significantly reduced. This energy reduction increases when the number of domains increases. 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

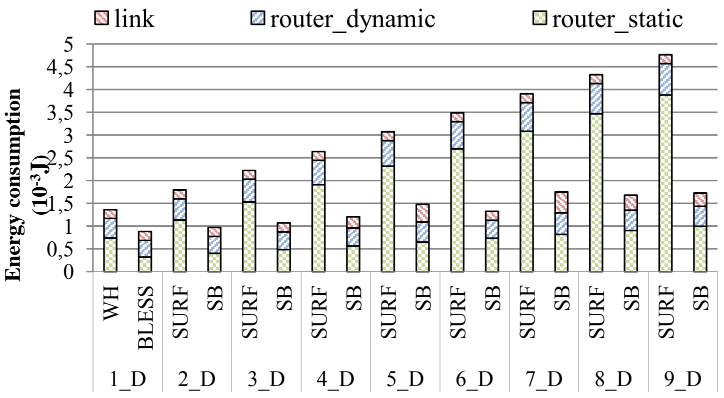


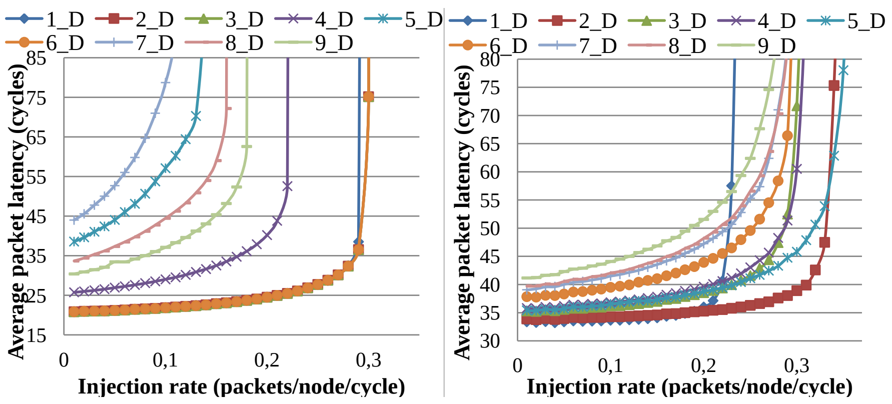
Figure 6.6: Energy consumption across different number of domains.

input ports need to increase the number of VCs to separately store packets for different domains. In SURF, all of the input ports need to increase the VCs to separately store packets for different domains, which causes significant increase of the static energy consumption. On the other hand, compared with BLESS, as our SB needs multiple VCs in the injection input ports, SB causes higher static energy consumption than BLESS.

It should be noted that the dynamic power consumption does not increase linearly with the number of domains for SB. This is because under different number of domains, the average packet latency is different, which will be introduced in Section 6.6.1. As the packet cannot stop in our SB, the higher average packet latency means that packets need to be transferred through more routers, which results in high dynamic power consumption.

Average Packet Latency under Different Number of Domains

Figure 6.7(a) and Figure 6.7(b) show the average packet latency of SB and SRUF under 1_D, 2_D,· · · , 9_D, respectively. In Figure 6.7(a), the curves for 2_D, 3_D, and 6_D are overlapped and have the lowest average packet latency. 4_D, 5_D, 7_D, 8_D, and 9_D have higher average packet latency. This is because, in our SB, packets of a domain can be transferred to the output ports only in the time slots allocated to the this domain. Sometimes, some packets of a domain have to be deflected because some output ports are not in the time slots allocated to the same domain. Therefore, compared with BLESS (1_D), our SB causes more packet deflections in 4_D, 5_D, 7_D, 8_D, and 9_D. However, such high number of packet deflections caused by the time slot allocation does not happen in 2_D, 3_D, and 6_D. This is because, based on



(a) BLESS (1_D) and SB (b) WH (1_D) and SURF

Figure 6.7: Latency across different number of domains.

the waves for 2_D, 3_D, and 6_D, the time slots of all of the output ports in a router are always allocated to the same domain. Thus, the time slot allocation in our SB does not cause extra packet deflections in 2_D, 3_D, and 6_D. Another reason resulting in the packet latency increase is that, in our 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 is significant for 4_D, 5_D, 7_D, 8_D, and 9_D, but this negative impact does not happen for 2_D, 3_D, and 6_D because all of the output ports in a router are always in the time slots allocated to the same domain.

Comparing Figure 6.7(a) with Figure 6.7(b), when the number of domains is 5,7,8, and 9, SB has higher average packet latency than SURF. This is because the packet injection/ejection happens only on the W_{SE} sub-waves in SB. Only when the sub-wave W_{SE} for a domains reaches the destination router, a packet can have a chance to be ejected. Otherwise, the packet has to be deflected. When the number of domains increases, each domain has fewer waves assigned to it. Packets in a domain have less chance to be ejected, but more chance to be deflected. As a consequence, the packet latency sharply increases in SB. In contrast, routers in SURF have VCs to temporarily store packets and do not need to deflect packets. Therefore, even though the packet latency still increases with the increase of the number of domains, the increase of the packet latency in SURF is much less than in our SB.

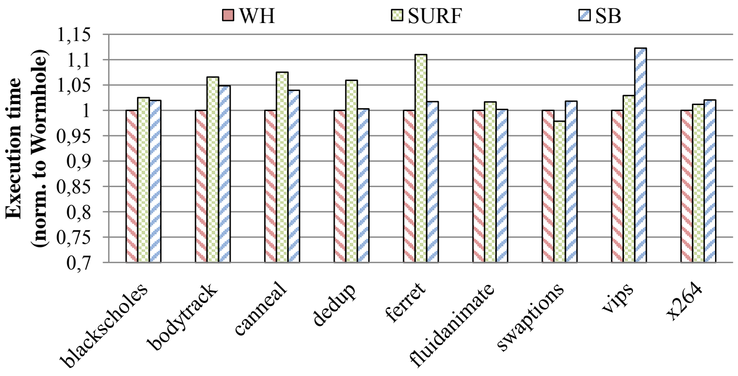


Figure 6.8: Application execution time.

6.6.2 Transfer of Multiple Class Packets

In a cache coherence protocol, there are multiple class packets, that should be separately stored in different VCs and transferred in different virtual networks [DT04]. It is necessary to guarantee that there is no protocol deadlock. However, as there are no VCs, the conventional bufferless NoCs does not support the transfer of multiple class packets. In contrast, by supporting confined-interference communication, our SB can easily support the transfer of different class packets for a cache coherence protocol.

To do so, we apply our SB on transferring different class packets for the MESI protocol [PP84], which needs two data virtual networks to transfer 5-flit packets and one control virtual network to transfer 1-flit packets. To separate the packets of the three different virtual networks, we set three domains and the packets of each virtual network are assigned/injected to one domain. There are 42 waves in our experiment. To continuously transfer 5-flit packets, the packets of one data virtual network are assigned to the waves $\{0, 1, 2, 3, 4, 15, 16, 17, 18, 19, 30, 31, 32, 33, 34\}$, and packets of the other data virtual network are assigned to the waves $\{7, 8, 9, 10, 11, 22, 23, 24, 25$ $26, 37, 38, 39, 40, 41\}$. In order to guarantee that the 5-flit packets in these two data virtual networks are transferred continuously, the head-flit of a 5-flit packet can be transferred only on the waves $\{0, 15, 30\}$ and $\{7, 22, 37\}$, respectively, and the rest of flits follow the head-flit. The 1-flit packets of the control virtual network are assigned to the rest of the waves, i.e., $\{5, 6, 12, 13, 14, 20, 21, 27, 28, 29, 35, 36\}$. As BLESS does not support the transfer of multiple class packets with various number of flits, BLESS is not considered in this experiment.

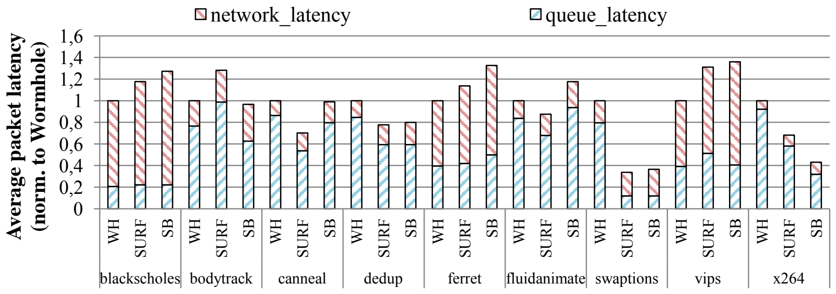


Figure 6.9: NoC packet latency.

Application Execution Time

Figure 6.8 shows the execution time of nine applications from the Parsec [BKSL08] benchmark suite, where the execution time is normalized to WH as the baseline (each input port has two data VCs and one control VC to build three virtual networks). 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 benchmarks dedup and fluidanimate, our SB achieves less than 1% performance penalty. In the vips benchmark, our SB has its largest performance penalty of 12.27%, whereas, for SURF, the largest performance penalty of 10.99% occurs for the ferret benchmark.

An interesting observation is that SURF achieves 2.14% performance improvement for the swaption benchmark. This improvement can be attributed to the acceleration of some packets' transmission. As shown in Figure 6.9, SURF has lower average packet latency than WH. Even though SB also has lower average packet latency than WH, our SB does not achieve such performance improvement. This is because the packet latency in SB is not significantly reduced as in SURF. Furthermore, most of the reduction of the packet latency in our SB and SURF comes from the control packets, which has a limited impact on the overall application performance. Only when the reduction of the packet latency is significant enough, SURF and SB can gain performance improvement.

NoC Packet Latency

Figure 6.9 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 in the NoC (network latency). Compared with WH, the average packet latency in SURF and our SB is significantly reduced in the benchmarks dedup, swaption, and x264, whereas, in benchmarks blackscholes, ferret, and

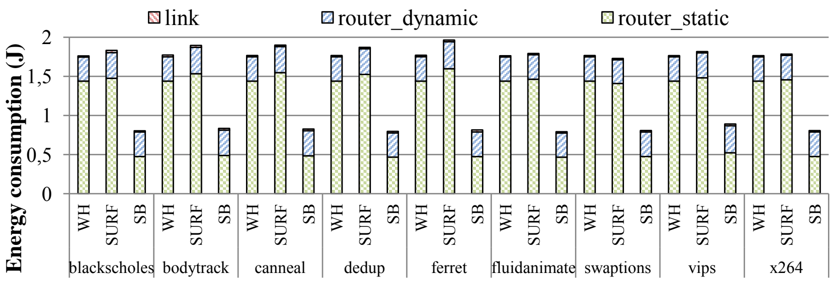


Figure 6.10: NoC energy consumption.

vips, the average packet latency increases. Most of the latency decrease comes from the queue latency reduction for the packets, whereas the latency increase is caused by the increase of the network latency. This is because based on the wave assignment in SURF and our SB, the network resources are reserved by different virtual networks to achieve confined-interference communication. However, since a domain can use the resources only in the time slots allocated to it, these resource reservation mechanisms cannot efficiently utilize the resource and undermine the efficiency of the networks, which causes the network latency increase.

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 is not essential to improve the overall application performance. As a consequence, even though the average packet latency in SURF and SB is reduced for some applications (see Figure 6.9), SURF and SB still cause performance penalty in the application execution time as shown in Figure 6.8.

NoC Energy Consumption

Figure 6.10 shows the NoC energy consumption for the nine applications. It is broken down into three parts, the static and dynamic energy consumption of routers and the total energy consumption of links. As can be seen in Figure 6.10, our SB consumes much less energy than the related approaches. Compared with WH, our SB reduces the total NoC energy consumption with 53.6% on average and compared with SURF, our SB reduces the total NoC energy consumption with 55.53% on average. This high energy reduction is achieved by eliminating the VCs for the N, S, W, and E input ports in a route for our SB to reduce the static energy consumption. In contrast, SURF has higher energy consumption than WH and our SB. This is because, with similar hardware structure of routers, WH and SURF have similar power consumption, but SURF needs more time to execute the applications. The link energy consumption is negligible because we use the low workload mode in the Parsec benchmark suite.

6.7 Discussion

In this chapter, we propose the Surf-Bless routing approach 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 approach achieves confined-interference communication with low energy consumption. The utilization of the wave pattern introduced in Section 6.5 is only suitable for a NoC with $N \times N$ 2D mesh topology. It is possible to extend our Surf-Bless to a $M \times N$ 2D mesh topology, but M and N must be the relation of integer multiple. It is also possible to extend our Surf-Bless to other mesh-based topologies, such as the 3D mesh topology and the concentrated mesh topology. However, it is not easy to extend our Surf-Bless to the tours topology, because the wire delay on a tours topology is not the same, which makes it difficult to construct the wave pattern. However, considering that NoCs with such topology are widely used in real chips, as shown in Table 1.1, our Surf-Bless routing approach is very promising and widely applicable.