A Deflection-Based Deadlock Recovery Framework to Achieve High Throughput for Faulty NoCs

Yibo Wu, Liang Wang, Xiaohang Wang, Jie Han, Shouyi Yin, Shaojun Wei, Leibo Liu

***Abstract*—Deadlock is a critical issue in faulty NoCs. Existing deadlock-free approaches on faulty NoCs suffer from low throughput and poor fairness when the network becomes over-saturated. This problem hinders their practical use as over-saturation scenarios are more frequent on faulty NoCs. To address this issue, a deflection-based deadlock recovery framework is proposed for higher over-saturation performance on faulty NoCs. First, we observe the low over-saturation performance of existing deadlock recovery approaches, and analyze the positive feedback loop that can amplify the negative impact of deadlocks and congestions, which necessitate handling both deadlocks and congestions in a deadlock recovery framework. Second, we propose a novel deadlock recovery framework, which includes an accurate, timely deadlock detection and a highly-efficient deadlock recovery. Both the deadlock detection and recovery reduce the average packet traversal latency, thereby improving the average over-saturation throughput. Third, we propose a distributed implementation to make the entire network enter and exit the deflection mode, which is conducted by broadcasting special messages via a bufferless subnetwork. An average over-saturation throughput improvement of 1.1$\sim$8.1$\times$ over state-of-the-art approaches is achieved. In terms of fairness, the minimal over-saturation throughput is improved from near zero to half of the peak throughput.**

***Index Terms*—faulty NoCs, deadlock recovery, deflection mode, over-saturation performance**

I. INTRODUCTION

As chips are designed with a growing number of cores for better parallelism, Networks-on-Chip (NoCs) have been considered to be promising on-chip interconnections due to their better bandwidth and scalability characteristics [11]. Aggressive transistor scaling makes it easier for chips to suffer from failures [2], [5], [44]. Therefore, future NoCs are likely to be irregular [47], which makes the deadlock a severe problem because conventional solutions for deadlock freedom (e.g. deterministic routing) applicable on regular NoCs are no longer effective. To achieve deadlock freedom in faulty NoCs, deadlock avoidance based [2], [40], [44], deadlock recovery based [46], [47] and subactive [36]–[39] approaches have been proposed [12]. But they all suffer from performance penalties especially when the network becomes over-saturated, resulting in a waste of on-chip resources.

Deadlock avoidance based approaches such as Immunet [44], uDIREC [40] and ARIADNE [2], proactively avoid deadlock dependence by placing some turn or virtual channel (VC) usage restrictions in the network. But these restrictions reduce the saturation point and limit the over-saturation throughput. In contrast, deadlock recovery based approaches such as Static Bubble [47], SPIN [46], allow deadlocks to form but develop mechanisms to detect and recover from deadlocks. There is a third type of approaches such as BBR [37], BINDU [39], DRAIN [36] and SWAP [38]. They claim themselves to be subactive [36] because they do not detect deadlocks but relies on the periodic packet movement to resolve deadlocks. Despite achieving a higher saturation point, both recovery-based and subactive approaches cannot maintain a high throughput as well as a good fairness when the network is over-saturated. The main reason is that they fail to detect deadlocks timely or recover from deadlocks with a high efficiency. These two weaknesses incur a long average flit traversal latency and degrade the over-saturation throughput.

On the other hand, faulty NoCs are more easily saturated because of fewer available bandwidths and reduced path diversity. Therefore, although realistic benchmarks exhibit relatively low injection rates, over-saturation scenarios in faulty NoCs appear more frequently than in a regular mesh NoC. According to our experiments conducted on an 8$\times$8 mesh network and under bit complement traffic, 20 faulty links can reduce the saturation point of ARIADNE by nearly 60%. As a result, it is necessary to develop a deadlock-free approach to achieve higher throughput and better fairness in an over-saturated faulty NoC.

In this paper, DeDR, a **De**flection-Based **D**eadlock **R**ecovery framework is proposed to achieve high over-saturation throughput and fairness for faulty NoCs. DeDR is based on an observation of a positive feedback loop between deadlocks and congestions. After timely detection of a potential deadlock, DeDR makes the network enter a deflection mode in a

This research program was supported in part by the National Natural Science Foundation of China (Grant No.61834002), and in part by the National Key R&D Program of China (Grant No. 2018YFB2202101), and in part by the National Science and Technology Major Project of the Ministry of Science and Technology of China(Grant No. 2018ZX01027101-002), and in part by the Natural Science Foundation of Guangdong Province under Grant 2018A030313166, and in part by Pearl River S&T Nova Program of Guangzhou under Grant 201 806010038, and in part by Fundamental Research Funds for the Central Universities under Grant 2019MS087, and in part by Open Research Grant of State Key Laboratory of C omputer Architecture Institute of Computing Technology Chinese Academy of Sciences under Grant CARCH201916, and in part by National Natural Science Foundation of China under Grant 61971200. (*Corresponding author: Leibo Liu.*)

Yibo Wu, Shouyi Yin, Shaojun Wei and Leibo Liu are with the Institute of Microelectronics, Tsinghua University, Beijing, China. Leibo Liu and Shaojun Wei are also with Beijing National Research Center for Information Science and Technology, Beijing, China. Email:wyb18@mails.tsinghua.edu.cn , {yinsy, wsj, liulb}@tsinghua.edu.cn

Liang Wang is with the School of Computer Science and Engineering, Beihang University, Beijing, China. Email: lwang20@buaa.edu.cn

Xiaohang Wang is with the School of Software Engineering, South China University of Technology, Guangzhou, China. Email: xiaohangwang@scut.edu.cn

Jie Han is with the Department of Electrical and Computer Engineering, University of Alberta, Canada. Email: jhan8@ualberta.ca

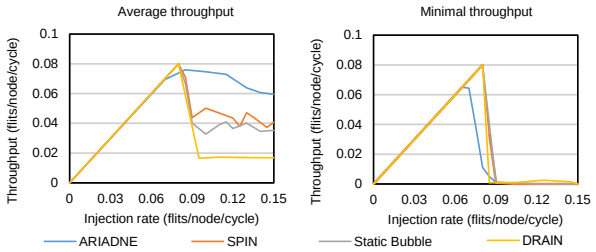


Fig. 1. Average and minimal throughput comparisons under bit complement traffic

distributed manner and handle both deadlocks and congestions with a high efficiency. By improving the deadlock detection timeliness and deadlock recovery efficiency, the average flit traversal latency is reduced, thereby improving the over-saturation throughput.

The contributions of this paper are:

• We observe significant over-saturation throughput degradation for existing deadlock recovery approaches. We also observe a positive feedback loop between deadlocks and congestions that can amplify the negative impact of these reasons for deadlock recovery based approaches. These analyses form the basis of DeDR design.

• We propose a deadlock recovery framework that handles both deadlocks and congestions in the same framework. DeDR judiciously combines the probe detection and timeout counters to achieve an accurate and timely deadlock detection. When detecting a potential deadlock, DeDR drives the network into a deflection mode, routes and ejects all packets in a backpressureless manner with a high efficiency. DeDR is topology agnostic, and achieves fully adaptive routing and deadlock freedom with minimally one VC for every message class.

• We present a distributed implementation to make the entire network enter and exit the deflection mode. The implementation is done by broadcasting a triggering message and an empty signal via a bufferless subnetwork, and using a one-bit wire grid to finish the deflection mode. The experimental results demonstrate that under real workloads, DeDR can reduce the average latency by up to 50%.

The rest of the paper is organized as follows. Section II details our motivation to improve over-saturation performance and the positive feedback loop between deadlocks and congestions. Section III describes how DeDR works. Section IV describes the required hardware overhead. Section V presents the experimental results. Section VI presents some related work. Section VII concludes the paper.

II. Motivation

We observe that existing deadlock avoidance based, deadlock recovery based and subactive approaches fail to maintain a high throughput as well as a good fairness when the network is over-saturated. To demonstrate the performance degradation, Figure 1 shows a case study to compare the over-saturation throughput of four approaches: one representative avoidance-based approach, ARIADNE [2]; two representatives recovery-based approaches, SPIN [46] and Static Bubble [47]; one representative subactive approach, DRAIN [36]. We choose these four approaches and compare with them in Section V, because they are topology agnostic and achieve deadlock freedom with minimally 1 VC per message class. As the injection rate increases and the network becomes over-saturated, the average throughputs of these approaches significantly degrade and the minimal throughputs approach zero. In Figure 1, the average throughput is defined as the average rate for flits to be accepted at destinations. The minimal throughput is defined as the lowest flit accept rate among all traffic flows [12]. A minimal throughput close to zero indicates a poor fairness in network bandwidth allocation. The reasons for the performance degradation are analyzed below from aspects of saturation point, over-saturation throughput and fairness. Particularly, for deadlock recovery based approaches such as Static Bubble, SPIN and DRAIN, there exists a positive feedback loop between deadlocks and congestions that can further worsen the problem.

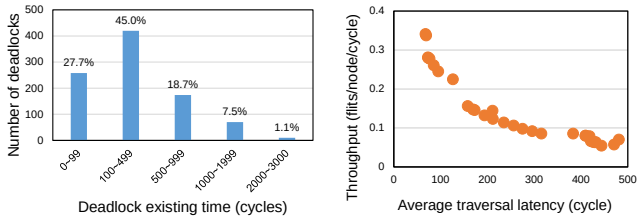
A. Saturation Point

The limited average over-saturation throughput of ARIADNE is restricted by its low saturation point. ARIADNE is based on spanning tree construction and uses up*/down* [48] turn restrictions to avoid deadlocks. The routing restrictions lead to non-minimal routing and reduced path diversity [47], and consequently reduce the saturation point. This problem is prevalent in other deadlock avoidance based approaches [2], [19], [29], [32], [40]. Deadlock recovery based approaches typically do not suffer this problem because they remove the routing restrictions. Their fully adaptive routing can make better use of the network bandwidth.

B. Over-saturation Throughput

Although SPIN, Static Bubble and DRAIN achieve a higher saturation point than ARIADNE, they have a dramatic average throughput degradation when the network is over-saturated. Both Static Bubble and SPIN use a probe detection mechanism which has a low false positive rate for deadlock detection. For deadlock recovery, Static Bubble augments a subset of routers with additional bubble buffers, so that Bubble Flow Control [43] can be enabled for recovery. SPIN recovers from deadlocks by orchestrating all routers in the deadlock dependence chain to synchronously transmit their stalled flits. DRAIN does not attempt to detect deadlocks but relies on periodically draining packets along an escape ring cycle to recover from deadlocks. The main reasons for the over-saturation throughput degradation of these three approaches lie in the deadlock detection timeliness and deadlock recovery efficiency. They are detailed as follows.

Deadlock detection timeliness: Both Static Bubble and SPIN adopt the probe detection mechanism [46], [47], [55] which often seriously delays deadlock detection. The mechanism works as follows. When a router detects a packet being stalled over a threshold time, the router sends a probe from this stalled packet. The probe follows routers along the buffer dependence chain and will return to the sender router if there is a deadlock. Then, the sender router confirms a



(a) Histograms of deadlocks' existing time in SPIN (b) Average over-saturation throughput and average flit traversal latency

Fig. 2. Motivational examples

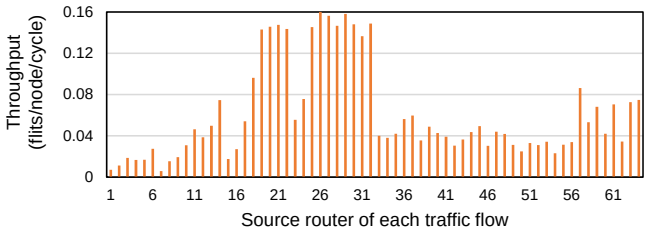


Fig. 3. Throughput of every traffic flow in SPIN under bit complement traffic

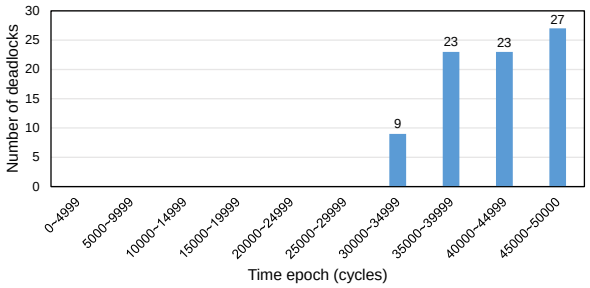


Fig. 4. Number of observed deadlocks in Static Bubble

deadlock. However, we observe that the probe mechanism fails to guarantee timely deadlock detection, which can impact over-saturation throughput. Figure 2(a) shows the existing time of deadlocks in SPIN until they are detected. In the case study, the total number of detected deadlocks is over 900, but only 27.7% of them are detected within 100 cycles. Some deadlocks are even detected after they have formed for more than 1000 cycles. This phenomenon happens due to the probe priority and dropping. It is possible that one probe sent out due to a real deadlock has a lower priority, whereas another probe sent out due to congestion has a higher priority. The less prioritized probe would be dropped. Therefore, the effective deadlock detection that can detect a real deadlock is interrupted, and the deadlock is not timely detected. Because DRAIN does not attempt to detect deadlocks, its periodic packet movement is often late in resolving deadlocks and renders itself suffer from the problem of deadlock detection timeliness.

Deadlock recovery efficiency: The second reason is the low deadlock recovery efficiency. It is observed that with the same network configurations of Figure 2(a), SPIN requires 47 cycles[1] on average to address one deadlock after deadlock detection. Considering the total number of deadlocks, which is more than 900 during the 100K-cycle runtime, for more than 40% of the runtime, a deadlock exists and the network is trying to recover from it. The deadlock forms a hotspot in the network and causes serious tree saturation [12], [42]. Packets enroute are very likely to encounter deadlocks and congestions. Furthermore, the recovery procedures only handle the deadlocks, leaving the network still highly congested after recovery. These two factors lead to high flit traversal latency. DRAIN periodically routes packets one hop forward along the draining cycle and claims that deadlocks are handled subactively with little effort. If the frequency of draining is high, many packets are forced to be detoured and the traversal latency would increase. If the frequency of draining is low, the network would be obtuse in reacting to deadlocks. Many packets would be stalled in the network waiting for the deadlock to be resolved, which also significantly increases the traversal latency.

We note that the average over-saturation throughput is generally reciprocal to the average flit traversal latency, as shown in Figure 2(b). This figure is obtained from experiments on several deadlock freedom approaches for faulty NoCs with various per-hop latency and synthetic traffic patterns. Therefore, the over-saturation throughput degrades due to the high traversal latency which is induced by the low deadlock recovery efficiency.

C. Fairness

State-of-the-art deadlock-free approaches in faulty NoCs also incur serious fairness issues. In Figure 1, the minimal throughput among all traffic flows approaches zero when the network becomes over-saturated. The minimal throughput is defined as the lowest flit accept rate among all traffic flows [12]. It corresponds to the traffic flow that has the lowest accept to generate rate[2] and stands for network fairness [12]. The results of Figure 1 indicate that these four approaches fail to provide satisfying fairness. Figure 3 plots the throughput (accept rate) of every traffic flow in SPIN. The assigned synthetic traffic pattern is bit complement so that there are in total 64 traffic flows between different pairs of source routers and destination routers. Although all traffic flows are generated with the same rate, Figure 3 demonstrates that in SPIN, some traffic flows sustain a high throughput while some are seriously starved, with a minimal throughput close to zero. Other approaches also generate similar trends to SPIN. A fair network should serve different traffic flows equally and generate higher minimal throughput. In real applications, once the network using SPIN saturates, some processors will hardly have a chance to maintain the operation due to the starvation of their traffic flows [12], [25].

D. Positive Feedback Loop between Deadlocks and Congestions

We observe that for deadlock recovery based approaches, there exists a phenomenon that can further amplify the negative impact of aforementioned weaknesses. Once a deadlock

[1]We assume that special messages used in SPIN and Static Bubble take one cycle for each hop.

[2]All traffic flows are generated with the same rate in synthetic experiments in this paper.

accidentally occurs, the network becomes over-saturated and deadlock recurrences are expected to be frequent. Figure 4 plots the number of observed deadlocks in Static Bubble under uniform random traffic. The simulation runtime is 50K cycles and the first deadlock is detected at the 32406th cycle. Before the detection of the first deadlock, the network is slightly congested but not saturated yet. However, after the handling of the first deadlock, deadlocks occur with an average frequency of 4.6 times/thousand cycles.

This phenomenon can be explained by a positive feedback loop between deadlocks and congestions. After a deadlock is detected and the network is recovering from it, routers in both the deadlock chain and adjacent congested area become hotspots of the network, with every VC taken up and preventing further injection from their source queues. Hence, packets to be injected are accumulated in the source queues. After the deadlock recovery, injection resumes and packets are injected at the highest rate that the number of available credits and free VCs allows. Besides, most of previous approaches only address the deadlock without paying attention to adjacent congested regions. These two facts render the network seriously congested again. According to the formation of deadlocks [12], [23] and previous observation [46], [47], when the network is congested with a high VC utilization rate and a high injection rate, the network is prone to deadlock. Consequently, deadlock recurrences are expected to be frequent after the first deadlock occurs accidentally, and the network also becomes over-saturated.

DeDR is not proposed to mitigate the positive feedback loop. Instead, DeDR makes use of the characteristics of the feedback loop to improve the over-saturation throughput. DeDR detects the first deadlock accurately so as not to unnecessarily fall into the feedback loop. Once a deadlock occurs, DeDR manages to detect the subsequent deadlocks in a timely manner and recover from them with a high efficiency.

III. Proposed Approach

Rather than proactively avoiding deadlocks, DeDR detects deadlocks and then manages to recover from them. During the deadlock recovery procedure, the network enters a deflection mode, in which packets are routed in a backpressureless manner [34]. The implementation of the deflection mode is carried out with some special messages in a distributed manner.

In this paper, we focus on link failures that render the network irregular [2], [45], [47]. Other faulty network components can be treated as link failures. For example, a faulty output port is the same as disabling the link connected with this port. Besides, like previous work [2], we assume that link failures are permanent and use an ideal fault detection method.

A. Deadlock Freedom Framework

1) Deadlock detection:

Based on the deadlock detection timeliness analysis in Section II-B and the positive feedback loop in Section II-D, we claim that a deadlock detection mechanism should be both accurate and timely. In DeDR, the probe detection [46], [47] and timeout counters [3], [10] are combined to achieve this goal. The probe detection detects deadlocks accurately and has a low false positive rate. However, it fails to detect deadlocks in a timely manner. In contrast, timeout counter never misses any deadlocks and guarantees that all deadlocks can be detected timely once they have existed over a predefined threshold time. However, timeout counter sometimes treats false positives as deadlocks.

According to the positive feedback loop, the first deadlock should be detected accurately with a low false positive rate, otherwise the network might unnecessarily fall into the loop and the saturation point would be reduced. Therefore, in DeDR, when an application starts operating, the first deadlock is detected with probe messages.

The recurrent deadlocks should be detected timely, so that the over-saturation throughput would not be degraded. But accuracy is not very important for them, because deadlock recurrences are expected to be frequent. Therefore, after recovering from the first deadlock, DeDR turns to timeout counters for future deadlock detection. The timeout counters mechanism confirms a potential deadlock once a packet is found to have been stalled in the current router over a threshold time. If deadlocks have not been detected by timeout counters for a long period, e.g. 10K cycles, which implies that deadlocks are unlikely to occur in a short period and the positive feedback loop is broken, the probe detection mechanism is again enabled for the next deadlock detection. By judiciously combining the probe detection mechanism and timeout counters, the deadlock detection of DeDR is both accurate and timely.

We note that Static Bubble and SPIN [46], [47] require the probe to accurately locate the deadlock for subsequent recovery procedures, which prevents using timeout counters and hurts their scalability. It is estimated in Static Bubble that a 128-bit wide probe can record up to 59 turns in an 8×8 mesh network. First, theoretically a deadlock chain can cover up to 64 nodes and be 63 hops long [2], in which case the probe message would fail. Second, the probe mechanism has a limited scalability. On a 16×16 mesh network, the scale of deadlock chains can be larger and wider links are needed to record longer probe paths. But different from Static Bubble and SPIN, DeDR only needs to confirm the existence of a deadlock rather than accurately locating where it is. Therefore, timeout counters can also be used for timely detection of deadlocks. And the probe message here does not need to record the paths so that there is a better scalability.

2) Deadlock recovery:

To improve the deadlock recovery efficiency, we design a deflection mode to handle both deadlocks and congestions in the deadlock recovery framework. The deflection mode mainly has two differences compared with the normal state. First, packet injection is forbidden unless the head flit of a packet has already been injected before the start of the deflection mode. Second, in the deflection mode, flits in the network are routed in a backpressureless manner, which is similar to that of bufferless deflection [15], [16], [34]. The backpressureless flow control is intrinsically deadlock free and is used to drain all packets remaining in the network.

During the deflection mode, flits en route are not buffered

anywhere. When there is no flit on one incoming link, one randomly chosen flit buffered in VCs connected with this link is allowed to attend output port allocation and starts its routing in the backpressureless manner. Other flits remain in VCs until the link is empty again in the following cycles. Flit-by-flit routing is used so that probabilistic livelock freedom is guaranteed [12], [22], [27]. When a flit attends output port allocation but fails in the contention for the productive output port, instead of being dropped or waiting in buffers, the flit is deflected to another available output port, regardless of whether this output port brings the flit closer to its destination. The deflection mode finishes when all the packets in the network are ejected. Then, the credit count of every port is set to the maximum and the network returns to the normal states.

To provide routing deadlock freedom and guarantee that no packet is stalled or dropped anywhere, for every router, the number of input ports must be equal to that of output ports [34], so that there is always a maximal matching between input requests and output ports when deflection is allowed. This condition is satisfied if links of the topology and link faults are bidirectional. Many commonly used topologies such as Mesh and Torus satisfy the first requirement [34]. Moreover, if a fault disables an input or an output port and renders the link unidirectional, the corresponding output or input port is also disabled to make this link fault bidirectional [17].

The above descriptions detail how DeDR works. Similar to SPIN and Static Bubble, DeDR is a deadlock recovery framework for faulty NoCs rather than a complete NoC fault tolerance mechanism. It is plug-and-play with existing fault-tolerant routing algorithms [2], [17] to support deadlock-free fault tolerance. In the evaluations, we combine DeDR with routing tables of ARIADNE [2] (but the up*/down* routing restrictions used for deadlock freedom are removed).

B. Distributed Implementation of Deadlock Recovery

DeDR uses a triggering message for triggering of the deflection mode, an empty signal for confirming whether the network has become empty and a one-bit wire grid to finish the deflection mode. We claim that the implementation is fully distributed. We define 'fully distributed' as: (1) routers involved in the implementation only need local knowledge [2], (2) any failed implementation component will not incur global failure of the deadlock recovery process [17]. To satisfy the first definition, DeDR requires every router to be involved in the implementation, but every router only needs the mode and direction information of itself for the execution. To satisfy the second definition, if any implementation component of a router fails, the router can be treated as a totally failed router, and other routers can still initiate the implementation. In this section, the implementation of deadlock recovery is described from aspects of how to trigger and finish the deflection mode. We also illustrate the advantages of the deflection mode.

1) Triggering of the deflection mode:

The triggering of the deflection mode is carried out by broadcasting a triggering message. Starting from the router that detects a deadlock (we will denote it as DR, i.e. detector router, hereafter), this router enters the deflection mode and sends the triggering message to adjacent routers. After receiving the triggering message, a router enters the deflection mode, stall the allocation for the current cycle and forwards the triggering message to its adjacent routers. In an $N \times N$ mesh network, the triggering procedure takes at most (N^2-1) cycles to complete. When the network is free of faults, it takes at most 2(N-1) cycles. The transmission of triggering messages uses a separate subnetwork and takes one cycle for each hop. The subnetwork helps avoid link usage conflicts with normal flits during the triggering process and then is also used for header information transmission because the deflection mode uses flit-by-flit routing to avoid livelocks.

One possible concern is the buffer overflow problem due to the coexistence of routers in the deflection mode and routers in the normal state when transmitting the triggering messages. However, this can be avoided because if router A, which is already in the deflection mode is sending a flit to neighboring router B, which is not in the deflection mode, then router A must have sent a triggering message to router B simultaneously. The triggering message can reach router B earlier than the flit and ensure that when the flit reaches router B, router B has entered the deflection mode and is routing flits in a backpressureless manner.

Because a deadlock can influence several routers, it is likely that several routers detect deadlocks simultaneously and all become DRs. However, it is guaranteed that in the deflection mode, there is only one DR or the finish of the deflection mode might not be carried out correctly. To cope with the situations of multiple DRs, when a router receives several triggering messages simultaneously, there is a priority comparison among these messages to drop the less prioritized ones. A triggering message carries the DR id and the deflection mode start time of the DR. A message with an earlier DR deflection mode start time is prioritized over one with a later start time. If both messages have the same DR deflection mode start time, then the one with a smaller DR id is prioritized. If a message has a lower priority, the message is dropped. The router records the DR id and DR start time of the message with the highest priority in a small buffer and forwards this message to its adjacent routers. If a router already in the deflection mode receives a message with a higher priority, the router should update its recorded information and forward this message to adjacent routers.

A tree structure is constructed during the triggering procedure. This serves the purpose of the network empty confirmation, and its usage is detailed in the next section. Similar to ARIADNE [2], when the triggering message is forwarded, each router marks the directions as either up or down. An up direction brings the router closer to the DR, while a down direction brings the router away from the DR. However, instead of using this for turn restrictions such as in ARIADNE, the classification of directions is used to record how the triggering message is broadcasted from the DR to the entire network. The DR only has down directions. The routers that only have up directions are denoted as the farthest router (FR) hereafter.

2) Finish of the deflection mode:

Before finishing the deflection mode, the DR needs to confirm that all packets in the network are ejected and the network is empty. The up/down tree structure constructed during the triggering procedure is used for the DR to gather empty information from other routers. Every router has a one-bit register, whose value is set to true when there is no flit in the router. The register values of all the routers are logical ANDed and backpropagated to the DR via the tree structure. The empty information backpropagation starts from FRs when their register values are true. If a router receives empty signals from all its down directions, and its register value is also true, then it continuously sends an empty signal out of its up directions every cycle. The empty signal is also transmitted via the bufferless subnetwork and takes one cycle for each hop.

Because the one-bit register value is used only when the empty signal reaches the router, it is possible that a flit is on the connected link and the empty signal observes a register value of false. Then the flit might not be detected and the empty confirmation process is not accurate. To avoid such cases, for a router with one-stage pipeline [41], we regulate that the register value is set true if the router has been free of flits for three consecutive cycles. In this way, flits that are in three places, i.e. the input links, the routers and the output links, can all be detected.

When the DR receives empty signals from all its down directions and its register value is also true, the network is determined to be empty. Then the DR places a voltage pulse on a one-bit wire grid to inform all routers to finish the deflection mode. For a moderate network size (e.g. 8×8 Mesh), we assume that the voltage pulse can reach all routers along the same dimension [28]. The finish process is assumed to take 5 cycles to complete, which is short compared with the timeout counter threshold time (50 cycles in Section V). Therefore, the voltage pulse has a negligible impact on the next deadlock detection.

The deflection mode implementation shows a good scalability, even though the entire network needs to be driven into the deflection mode and the deflection mode needs to drain all packets remaining in the network. It is observed that under uniform random traffic, the deflection mode lasts 120 cycles on average in a regular 8×8 mesh network and 320 cycles in a regular 16×16 mesh network. Considering the quadratically increasing number of routers, a good scalability on larger networks can be expected.

3) Advantages of the deflection mode:

The first advantage is high recovery efficiency in the deflection mode. Although the injection is forbidden, the throughput remains high and the deflection mode does not influence the overall performance. In one case study under uniform random traffic, the average throughput of DeDR in the normal state is 0.303 flits/node/cycle and the average throughput in the deflection mode is 0.237 flits/node/cycle. This provides a higher average over-saturation throughput compared with other deadlock recovery based approaches, where the injection is allowed but the throughput during the recovery process is fairly low. In DeDR, the average duration of the deflection mode is 120 cycles, and all packets can be ejected within this time span. In contrast, after deadlock detection, SPIN on average requires 47 cycles to address just one deadlock. During this period, only the deadlock dependence circle is broken while few stalled packets are ejected. As for DRAIN, although it can also handle false positives in the recovery process (by draining packets), it only guarantees the proceeding of some packets that are likely to be involved in deadlocks. After one recovery, deadlocks may still exist and more recoveries are still required. Most packets are stalled in the network and few can be ejected during the recovery process. Therefore, its recovery efficiency is evidently lower than that of DeDR.

The second advantage is that every packet in the network is guaranteed to be ejected after the deflection mode. DeDR benefits from it in terms of fairness. In ARIADNE, SPIN, Static Bubble and DRAIN, the network is never free of packets in over-saturation scenarios. Depending on the assigned traffic pattern, some routers in the network are constantly involved in congestion. Packets in source queues of these routers have to undergo many allocations before they are injected. Therefore, many traffic flows from these routers are starved and the fairness problem worsens. In contrast, after the deflection mode, DeDR leaves a network free of packets, and all routers have the same chance to inject packets stalled in source queues. This significantly improves fairness.

C. Additional Implementation Details

1) Protocol deadlock freedom:

In cache coherent systems [51], the network using DeDR is required to separate different message classes and achieve protocol deadlock freedom. In the normal states, DeDR uses different virtual networks (VNets) for different message classes to avoid protocol deadlock.

During the deflection mode, DeDR statically time-multiplexes the links and crossbars between different time domains [54] to avoid protocol deadlocks. Each message class can only use its designated time domain so that different message classes are separated. For example, assuming that there are two message classes, request packets and reply packets. Reply packets are the terminating message class [56]. DeDR uses two time domains to separate these two message classes. During the first time domain, request packets can use all routing resources while reply packets are forbidden. During the second time domain, reply packets can use all routing resources while request packets are forbidden. The injection of reply packets is also permitted.

Proof of protocol deadlock freedom: During the deflection mode, protocol deadlocks occur when both the two conditions below are met and form a cyclic dependence. First, the network is full of packets belonging to nonterminating message classes (e.g. request packets) [56]. They prevent packets of terminating message classes (e.g. reply packets) from being injected into the network. Second, because packets of terminating message classes cannot be injected, the injection queues for terminating message classes are full and the packet generation of terminating message classes is prevented. Hence, packets of nonterminating message classes cannot be ejected and processed and they might fill the network. In DeDR, by

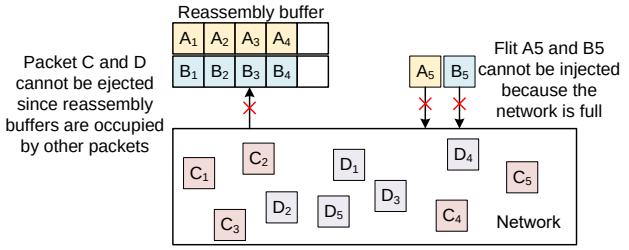


Fig. 5. Reassembly deadlock in bufferless deflection

statically time-multiplexing the links and crossbars between different time domains, packets of terminating message classes in the network are guaranteed to reach the destinations and be consumed. Therefore, packets of terminating message classes are guaranteed to be injected. In this way, the cyclic dependence is broken and protocol deadlocks are avoided in the deflection mode.

Although DeDR uses static time domains, the latency does not increase. This is because time domains are only used when routing deadlocks occurs and the network is usually deeply saturated with a long latency, the additional latency due to time domains is in fact minimal. By synchronizing the time domains of all routers during the triggering message broadcasting, DeDR can easily support protocol deadlock freedom during the deflection mode.

2) Packet reassembly:

Flit-by-flit routing is used in the deflection mode for probabilistic livelock freedom [12], [27]. Therefore, packets can be truncated and reassembly is necessary. A reassembly mechanism borrowed from [15], [22] is incorporated to address this problem. DeDR uses MSHRs that already exist in the private caches as the reassembly buffers. But the Retransmit-Once mechanism proposed in [15], which is used to avoid reassembly deadlocks, is eliminated.

In bufferless deflection, reassembly deadlocks occur due to the lack of buffer backpressure in reassembly buffers, as depicted in Figure 5. The reassembly buffers are occupied by some flits of packet A and packet B and cannot be released unless flit A_5 and B_5 are injected and enter the reassembly buffers. Flits constituting packet C and D cannot be ejected because reassembly buffers are occupied by other packets. However, flits A_5 and B_5 cannot be injected since the network is full of flits constituting packet C and D. There exists a cyclic resource dependence that needs to be broken.

According to the above descriptions, one way is to guarantee that for any half-injected packet, the remaining flits of this packet (e.g. flit A_5 and B_5 in Figure 5) outside the network will be injected later on. In this way, the fully-injected packet (e.g. packet A and B) can be successfully reassembled and release the reassembly buffers for other packets.

Proof of reassembly deadlock freedom: To satisfy the above goal, DeDR guarantees that: first, a fully-injected packet is always able to enter the reassembly buffers; second, there are always fully-injected packets in the network that can be reassembled and release buffers for further injection.

To satisfy the first requirement, we regulate that only tail flits can enter reassembly buffers and reserve space for the whole packet. If a tail flit is in the network, then the packet must have been fully-injected. In this way, it is impossible that reassembly buffers are occupied by half-injected packets as depicted in Figure 5.

The second requirement is naturally satisfied with typical VC configurations. It is usually assumed by NoC works that a VC has 4 or 5 flit-sized buffer entries and a data packet consists of 5 or 8 flits. According to this VC configuration, a data packet would require 1 or 2 VCs for buffering. Whenever a data packet is half injected, it would occupy one VC in the local input port. As can be derievd, if the network is able to provide 1 or 2 VCs for every half-injected data packets, namely, provide 1 or 2 VCs for every VC in the local input port, there must be some half-injected packets that will be fully injected later on. These fully-injected packets can thus be reassembled and release buffers for further injection. For a normal mesh network that has the same number of VCs in every input port, the number of VCs in nonlocal input ports is 3.5$\times$ more than that of the local input ports. Therefore, the second requirement is satisfied.

A corner case is that even if a flit has been injected into the network, it might be stored in buffers permanently. However, because DeDR randomly picks a flit for allocation and transmitting in every cycle, this case is unlikely to occur.

According to our experiments, reassembly deadlock is theoretically possible but never occurs. Therefore, our mechanisms incur little influence on the overall performance.

D. Necessity to Develop DeDR

Although one may argue that over-saturation throughput can be improved by adding VCs, applying congestion-aware adaptive routing or throttling mechanisms, these methods have difficulty completely erasing the negative impact from the deadlock-free approaches themselves. More VCs and adaptive routing [20], [31][3] only postpone the saturation point without addressing the aforementioned weaknesses. As the injection rate increases further, these methods still lose efficacy and the over-saturation throughput would decline dramatically. For example, we integrate DBAR [31] into SPIN's routing table, and still observe an average over-saturation throughput degradation of nearly 26% and a near zero minimal throughput under the network configurations of Figure 1. As for throttling mechanisms [6], [8], [35], [53], while they enable NoCs to achieve higher over-saturation throughput, the improvement is never ideal due to restrictions in their congestion awareness and throttling threshold adjustment.

First, throttling mechanisms require global congestion information gathering mechanisms to determine if the network is congested. However, there is usually a lag in the gathering and throttling decision making, damaging the accuracy of congestion estimation.

Second, to be applicable on different networks and under different workloads, throttling mechanisms should adaptively adjust the threshold of congestion estimation. A fixed throttling threshold can cause under-throttling or over-throttling under

[3]We note that the routing tables used in this paper already include congestion aware [2] adaptivity.

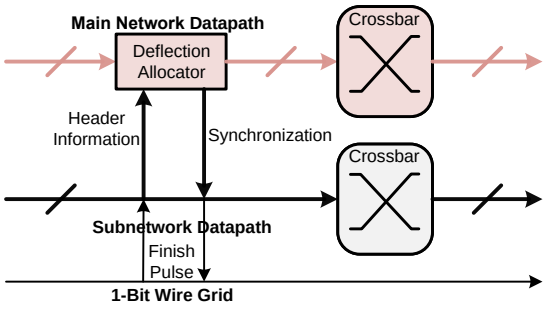


Fig. 6. DeDR router architecture overview

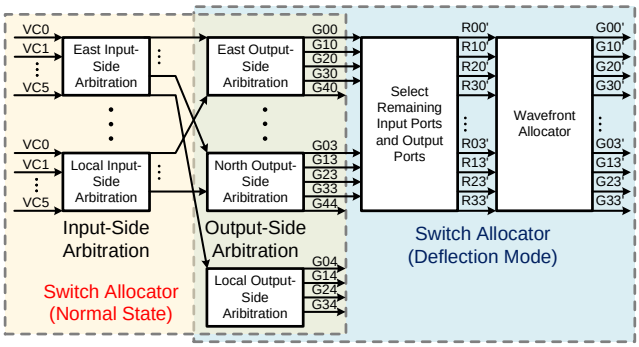


Fig. 7. Deflection allocator

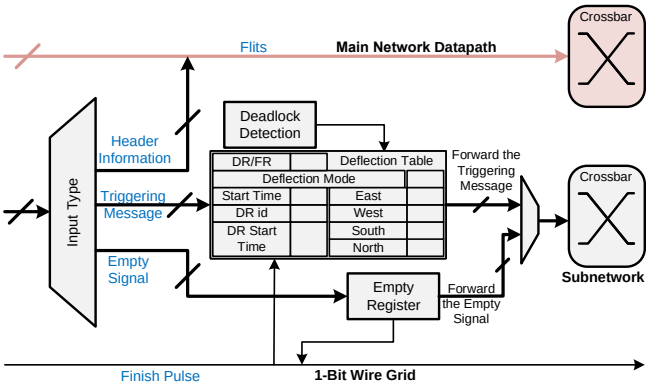


Fig. 8. DeDR subnetwork design

other network configurations or workloads. The threshold adjustment decision that is usually made according to the relationship between the threshold and throughput is not very accurate. Therefore, although throttling can reduce the probability of deadlocks and mitigate over-saturation degradation, the occurrence of deadlocks cannot be completely removed. Especially for deadlock recovery approaches with a low recovery efficiency, even occasional deadlocks and recovery can greatly impact the over-saturation throughput.

IV. Router Architecture

We modify a baseline router with a one-cycle router latency and a one-cycle link latency [41]. Figure 6 plots the router architecture overview. The three main changes are the deflection allocator, the subnetwork and one-bit wire grid. By extending a normal switch allocator, the deflection allocator enables deflections during the deflection mode. The subnetwork is used to transmit the header information and special messages during the deflection mode and stays synchronized with the main network. The one-bit wire grid is used to finish the deflection in a fast manner.

A. Deflection Allocator

Figure 7 shows how switch allocation in the normal state and the output port allocation in the deflection mode are addressed. We extend a separable input-first switch allocator with a selection unit and a wavefront allocator. In the normal state, switch requests are inputted into the input-side arbiters and the switch grants are generated by output-side arbiters, as the left yellow box shows.

In the deflection mode, switch requests are inputted into the output-side arbiters and the switch grants are generated by both output-side arbiters and the wavefront allocator, as the right blue box shows. The input side arbiters are eliminated, as DeDR only allows one randonly chosen VC to attend allocation during the deflection mode. However, to achieve protocol deadlock freedom, before entering the output side arbiters, each VC should be compared with the time domain to decide whether it can attend allocation. If a flit wins the contention for an output port, the switch grant is directly generated by the output-side arbiters. To enable deflections for flits that lose in contention, the selection unit is integrated in the allocator after the output side arbitration to select unoccupied output ports and requesting input ports without grants. Then, it generates new requests for deflections. A wavefront allocator is adopted after the selection unit to give random grants between these remaining input ports and output ports since the wavefront allocator ensures maximal matching. The final grants are derived by ORing the grants from the output-side arbiters and the grants from the wavefront allocator. Because DeDR usually disallows flit injection in the deflection mode and a packet cannot be deflected to a local output port if it has not reached the destination, a 4×4 wavefront allocator is sufficient for deflection. If a flit needs to be injected, it can be deflected to any output port that is currently not in use.

During the transition between the normal state and the deflection mode, the allocator always remains the same. The difference of the two modes is to decide where the switch requests should enter the allocator and where the allocator generates switch grants. This is easily done by adding several multiplexers and the router mode transition can occur within one cycle after receiving the triggering message.

B. Subnetwork

The subnetwork serves two purposes. The first is the generation and transmission of triggering messages and empty signals. The second is header information transmission for lookahead [41] and flit-by-flit routing [22], [27].

The triggering message should include the DR id and DR start time. The header information should include the message class, destination router id, flit number, and global MSHR

identifier for both routing computation and packet reassembly. A subnetwork with 16-bit wide links, buffers and crossbars should be enough.

Figure 8 shows the subnetwork design. The top red part corresponds to the main network[4] [41], and the gray parts are additional units including the subnetwork and one-bit wire grid. DeDR first decides the input type at every subnetwork input port. The input type can be header information, triggering messages and empty signals. The three bold lines starting from the input type unit in the left of Figure 8 represent the processing of each type of messages. The middle of Figure 8 shows a buffer that stores necessary information for the deflection mode and is denoted as the deflection table. The deadlock detection unit on top of the deflection table includes both timeout counters and the probe mechanism [47]. The empty register below the deflection table represents if the router has been free of flits for three consecutive cycles. Based on the input type, each router performs the following actions based only on local information.

- If it is a header information, it helps the flit in route computation and allocation.
- If it is a triggering message, the input type unit selects the triggering messages with the highest priority and sends them to the deflection table. If the router is not in the deflection mode, the router enters the deflection mode, records the message information in the deflection table, decides the up/down directions and forwards the message out of the down directions. If the router is in the deflection mode and the message has a lower priority, the message is dropped. If the router is in the deflection mode and the message has a higher priority, the router updates information in the deflection table, decides the up/down directions again and forwards this message out of the new down directions.
- If it is an empty signal, the signal is directly sent to the empty register. If there are empty signals from all down directions, the empty register value is 1 and the router is not the DR, the empty signal will be forwarded to all up directions. If the router is the DR, it places a voltage pulse on the one-bit wire grid to finish the deflection mode.

The subnetwork could increase the network power consumption if all its component are powered-on during the normal state. Besides, during the transition from the normal state to the deflection mode, all data packets need to copy the header information to the buffers of the subnetwork, which is difficult to complete within one cycle. Therefore, we make a modification on the triggering of the deflection mode. During the normal state, the buffers of the subnetwork are power gated [10] to save static power. When a router detects a deadlock, it first generates a voltage pulse on the one-bit wire grid to inform other routers to wakeup the subnetwork buffers and copy header information. After 10 cycles of wakeup (considering that 8 cycles are enough to wakeup a 128-bit wide router [9], 10 cycles should be a conservative assumption), the detector router starts broadcasting the triggering message on the subnetwork. During the 10-cycle wakeup, the network remains in the normal states.

[4]Except for the crossbar and input link, the other necessary components of the basic router are removed for brevity.

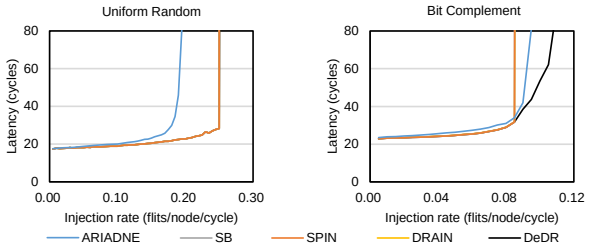


Fig. 9. A typical example of latency comparison. SB stands for Static Bubble

V. Evaluation

In this section, we present experimental results to illustrate DeDR's high over-saturation performance on faulty NoCs. All the experiments are performed using the gem5 [7] simulator with Garnet2.0 [1] on an 8x8 mesh network. Link faults are randomly inserted, but the network remains connected, i.e. at least one path exists between every pair of nodes. For every number of faulty links, we randomly generate enough different topologies until the average results stabilize. Each synthetic simulation runtime is 100K cycles, and the warmup time is 10K cycles. The simulation runtime under real workloads is 5M cycles. The flow control used is virtual cut through, although DeDR can also use wormhole. Each VNet has 2 packet-sized VCs. Each data packet consists of 5 flits and each control packet consists of 1 flit. The flit size is 128 bits. In synthetic traffic evaluation, the network has 1 VNet for data packets. In real workloads evaluation, the network has 3 VNets to achieve protocol deadlock freedom. The router model used has one-cycle router latency [41] and one-cycle link latency. DeDR is compared to ARIADNE [2], Static Bubble [47], SPIN [46] and DRAIN [36]. These four approaches are representative avoidance-based, recovery-based and subactive deadlock freedom approaches. For fairness of comparison, the deadlock recovery frameworks, DeDR, Static Bubble and SPIN, are combined with routing tables constructed in the same way as those of ARIADNE, except that the up*/down* routing restrictions are removed. DRAIN also uses the routing tables of ARIADNE when the network is not draining packets on the escape VC. But in practice, DeDR, Static Bubble, SPIN and DRAIN can also be combined with other routing methods [17], [18]. The deadlock detection threshold T of timeout counters is set as 40 cycles. The threshold time of the probe detection is set as 25 cycles. The drain epoch of DRAIN is set as 1K cycles. These configurable parameters are determined empirically.

A. Throughput and Latency

1) Synthetic traffic:

Figure 9∼11 shows typical examples of the latency, minimal throughput and average throughput comparisons under uniform random and bit complement traffic as the injection rate increases. The experiments are carried out on a network with a fixed distribution of 2 faulty links for performance demonstration.

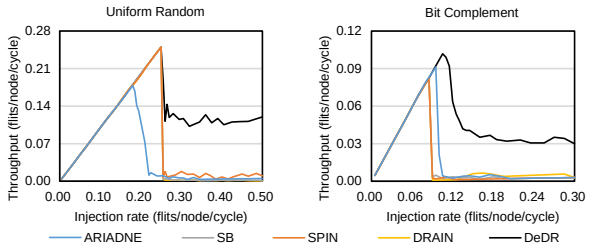


Fig. 10. A typical example of minimal throughput comparison

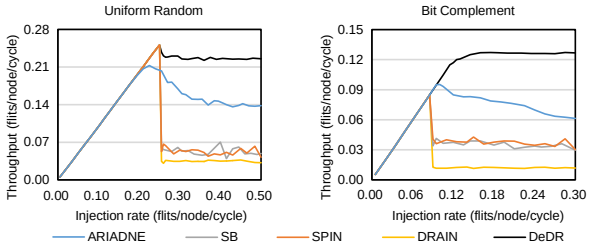


Fig. 11. A typical example of average throughput comparison

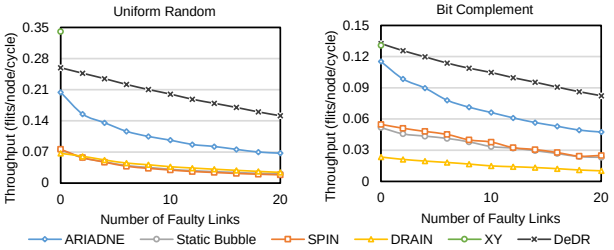


Fig. 12. Average over-saturation throughput comparison as the number of faulty links increases

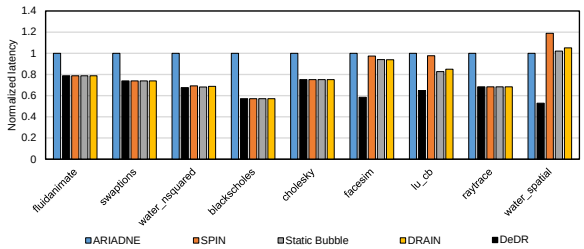


Fig. 13. Normalized latency in real workloads

In Figure 9, due to the removal of turn restrictions, DeDR, SPIN, Static Bubble and DRAIN have a slightly lower zero-load latency than ARIADNE. These four approaches also have the same saturation point which is higher than ARIADNE under uniform random traffic. This is because they benefit from the fully adaptive routing when no deadlock occurs. However, under bit complement traffic, the saturation point of SPIN, Static Bubble and DRAIN is the lowest because the network is much easier to suffer deadlocks and become saturated. The saturation point of DeDR is the highest because the occasional deflections help delay the network saturation.

In Figure 10, the minimal throughput of all approaches keeps increasing until the network becomes saturated. In the over-saturation scenario, DeDR is the only one that sustains a minimal throughput of almost half of the peak throughput. In contrast, the minimal throughput of other approaches all become close to zero. A network using DeDR would have a better fairness even if no additional fairness mechanism is applied.

In Figure 11, DeDR significantly improve the average over-saturation throughput of other approaches. The performance curves of SPIN and Static Bubble are almost indistinguishable due to the high similarity in their designs. DRAIN has the lowest throughput due to the inefficient draining path. The draining path is a single unidirectional ring that covers the whole network. Using a ring as the deadlock resolution method has been observed to be detrimental to throughput [44], [45].

Figure 12 shows the average over-saturation throughput comparisons as the number of faulty links varies. For every number of faulty links, DeDR consistently generates the highest over-saturation throughput. The improvements of DeDR over baselines is 1.1∼8.0× under uniform random traffic and 1.2∼8.1× under bit complement traffic. DeDR also generates the most graceful throughput degradation [12] as the number of faulty links increases, which is less than 40% for 20 faulty links. The throughputs of Static Bubble and SPIN are very close due to the similarity in their implementations. ARIADNE always has the second highest over-saturation throughput. But its performance degrades faster than DeDR as the number of faulty links increases. DRAIN has a slightly higher over-saturation throughput than SPIN and Static Bubble under uniform random traffic. But its throughput is the lowest under bit complement traffic. This is because the network is much easier to suffer deadlocks under bit complement traffic and DRAIN is slow in removing frequent deadlocks with the periodic forced packet movement.

In Figure 12, we also give the results when the network is fault-free and regular. Except for XY routing, the other approaches still use the adaptive routing tables constructed in ARIADNE and differ in the deadlock freedom mechanism. XY routing has a much higher over-saturation throughput than others under uniform random traffic due to its load balance. Under bit complement traffic, the over-saturation throughput of DeDR is slightly higher than XY routing. DeDR is more suitable for faulty networks. In regular networks, simpler approaches such as XY routing are more suggested to be adopted.

2) Real workloads:

Figure 13 shows the average packet latency under real workloads. The latency is normalized to ARIADNE. The workloads are adopted from Synfull [4]. The configurations of the workloads and cores in an 8×8 network are the same as [4], [30]. The network has a fixed distribution of 20 faulty links. In all 9 workloads, DeDR consistently has the lowest latency. Under water_nsquared, facesim, lu_cb and water_spatial, the network undergoes temporary deadlocks and saturations. DeDR is in the deflection mode for 5.45%, 11.63%, 4.13%, 8.90% of the runtime under these four wordloads repectively. Although deadlocks are rare in general, the positive feedback loop renders the latencies of SPIN, Static Bubble and DRAIN are higher than that of DeDR due to their lower over-saturation throughput and poor fairness. Although SPIN has a higher over-saturation throughput than Static Bubble under synthetic

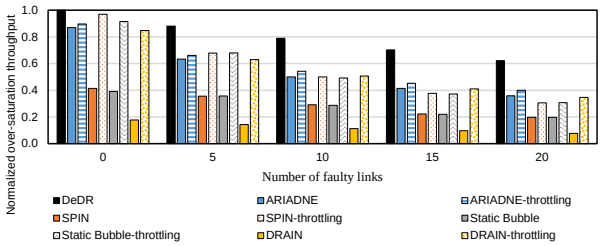


Fig. 14. Normalized average over-saturation throughput comparison with throttling-combined baselines

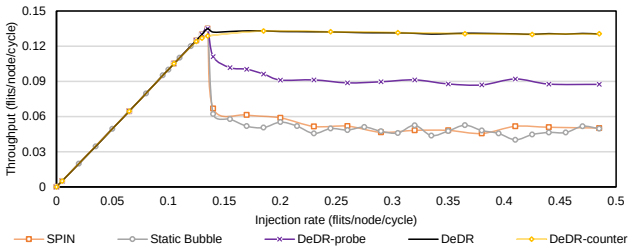


Fig. 15. Impact of deadlock detection methods

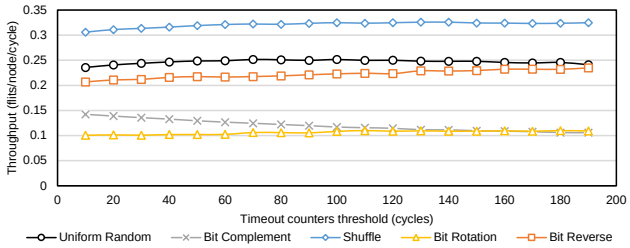


Fig. 16. Deadlock detection threshold sweep

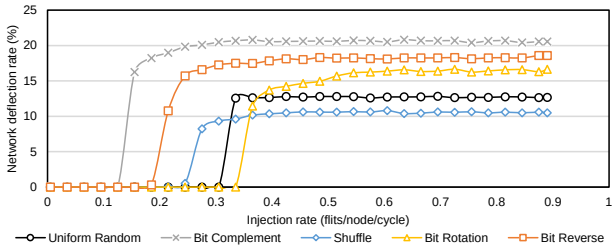


Fig. 17. Average network deflection rate

traffic, Static Bubble has lower latencies than SPIN across these 4 benchmarks due to fewer observed deadlocks. In the other benchmarks, the average injection rate is so low that deadlock never occurs during the runtime. Therefore, DeDR has the same performance as SPIN, Static Bubble and DRAIN. Due to the removal of up*/down* routing restrictions and the benefit of minimal routing, DeDR, SPIN, Static Bubble and DRAIN usually have lower latencies than ARIADNE.

B. Configurable Parameters

1) Comparison with throttling-combined techniques: Figure 14 shows the normalized average over-saturation throughput comparison when an additional throttling mechanism is combined with baselines. Columns in Figure 13 with the suffix '-throttling' represent baselines combined with an additional throttling mechanism. The throttling mechanism Tune [53] used for comparison achieves the best results among other previous works [6], [8], [35] and has been widely used for comparisons. Tune uses additional channels for buffer usage information gathering and congestion estimation. It can adaptively adjust the throttling threshold to suit the assigned traffic pattern and network configurations. All the experiments are carried out under bit complement traffic. It is assumed that 16 cycles are needed for a node to gather congestion information and the throttling mechanism never fails due to faults.

For all numbers of faulty links in Figure 14, the additional throttling mechanism can improve the over-saturation throughput, achieving an improvement of 1.03∼1.12× over ARIADNE, 1.56∼2.34× over SPIN and Static Bubble and 4.25∼4.79× over DRAIN. However, DeDR still consistently generates better results than the baselines, with an improvement of 1.03∼3.15×. The improvement of Tune over ARIADNE is not significant since ARIADNE has already achieved a relatively stable over-saturation throughput. The improvement of Tune over SPIN, Static Bubble and DRAIN is significant since throttling controls congestion and helps avoid most deadlocks. However, as explained in Section III-D, throttling fails to completely prevent deadlocks from occurring. Therefore, their low recovery efficiency can still lead to some performance penalties especially when the number of faulty links is high.

2) Different deadlock detection mechanisms: Figure 15 shows the average throughput comparison among SPIN, Static Bubble and DeDR using different deadlock detection mechanisms. The experiments are carried out in a regular network under bit complement traffic to illustrate the benefits of the combined deadlock detection method. DeDR-counter only uses timeout counters for detection, while DeDR-probe only uses the probe detection mechanism. The saturation point of DeDR-counter is 0.125 flits/node/cycle, whereas the saturation point of DeDR, DeDR-probe, SPIN and Static Bubble is 0.135 flits/node/cycle. The probe detection mechanism has a low false positive rate. Therefore, the latter approaches improve the saturation point of DeDR-counter by 8%. However, in terms of over-saturation throughput, DeDR and DeDR-counter achieve the best result, which is 1.4× higher than DeDR-probe, 2.5× higher than SPIN and 2.6× higher than Static Bubble. DeDR-probe performs worse than DeDR-counter as DeDR-counter ensures timely deadlock detection. But due to higher deadlock recovery efficiency of the deflection mode, DeDR-probe still generates better over-saturation throughput than SPIN and Static Bubble. By judiciously combining the probe detection mechanism and timeout counters, DeDR enjoys the benefits of both detection mechanisms.

3) Timeout counters threshold time sweep: Figure 16 sweeps the timeout counter threshold to study its impact on over-saturation throughput. The probe detection threshold is not considered here, as DeDR only relies on it for the first deadlock detection. Its value has a negligible impact on the over-saturation throughput. As the threshold increases, the throughputs under bit reverse, transpose and

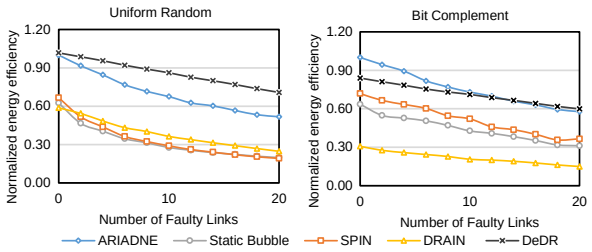


Fig. 18. Normalized energy efficiency under synthetic traffic

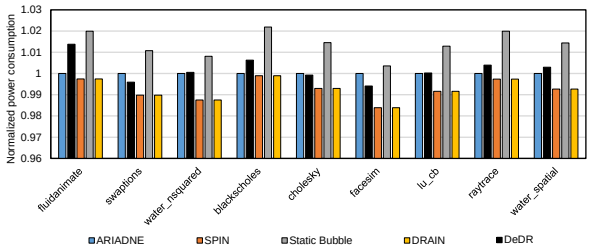


Fig. 19. Normalized power consumption under real workloads

shuffle increase as well. The throughput under uniform random does not vary much. The throughput under bit complement decreases a little. It can be concluded that a threshold value between 50 and 100 is suitable for most traffic patterns. However, a too large threshold value can lead to untimely deadlock detection, and the fairness problem could deteriorate, even if under most traffic patterns the average over-saturation throughput increases.

4) Average network deflection rate:

Figure 17 shows the average network deflection rate as the injection rate increases. The network deflection rate denotes the percentage of deflections among all switch grants. For all the synthetic traffic patterns in Figure 17, before a deadlock occurs, the deflection rate remains zero. Once a deadlock occurs, the network enters the positive feedback loop and the deflection rate increases. But as the injection rate further increases, the deflection rate stabilizes because the deflection mode controls the packet injection.

C. Power Consumption and Hardware Overhead

1) Power consumption:

Figure 18 shows the energy efficiency when the network is over-saturated and the results are normalized to ARIADNE. The energy efficiency is the reciprocal to the average energy consumed by each flit. The energy is calculated by gathering the runtime statistics and use DSENT [52] for estimation under 22 nm technology. DeDR consistently has the highest energy efficiency, with an improvement over baselines of up to 4.0× under uniform random traffic and 3.7× under bit complement traffic. Although DeDR consumes more energy due to the deflections and additional hardware, DeDR still has a higher energy efficiency due to its significantly higher over-saturation throughput.

Figure 19 shows the normalized power consumption comparison under real workloads. The power consumption of DeDR is quite close to ARIADNE, with an increment of up to 1.4% and a decrement up to 0.6%. Static Bubble performs slightly worse than SPIN, DRAIN and ARIADNE due to the additional bubble buffers. The average injection rate of the 9 workloads is low. Therefore, leakage power dominates the overall power consumption, making the gaps among different approaches rather small.

TABLE I
HARDWARE OVERHEAD COMPARISON

Approaches	DeDR	SPIN	Static Bubble
Additional units	allocator, subnetwork, wire grid, probe unit	SPIN fsm, probe unit, move unit, probe_move unit, kill_move	bubble buffer (21), Static Bubble fsm, probe unit, disable unit, check_probe unit, enable unit
Overhead	10.7%	4.0%	6.1%

2) Hardware overhead:

Table I plots the hardware overhead comparison among DeDR, SPIN and Static Bubble, with the results normalized to ARIADNE because all approaches in this paper use the routing tables adopted from ARIADNE. DRAIN is not compared in Table I because its hardware overhead is almost negligible. The middle of the table represents all the additional units we considered during implementation. The configurations of the baseline network is given in the first paragraph of Section V. The network frequency is 1GHz. By synthesizing the modified routers using the Design Compiler under 45nm TSMC library, the area of a baseline router is 105386 μm^2. The 21 additional bubble buffers is the main difference between Static Bubble and SPIN. Although DeDR only need the probe detection unit adopted from SPIN and Static Bubble, the additional units listed in Table I makes its overhead larger than Static Bubble and SPIN. However, considering the significant improved over-saturation performance, we believe that DeDR manages to achieve a good tradeoff between area and performance.

VI. Related Work

A. Deadlock Avoidance Based Approaches

Deadlock avoidance based approaches [2], [19], [29], [32], [40], [44] attempt to place routing restrictions or limit usage of other traffic resources such as VCs to guarantee that no deadlock occurs. Immunet [44] uses three routing tables for adaptive routing and switches to a deadlock-free escape VC[5] using BFC to avoid faulty links. However, a large hardware overhead and long latency [2] are incurred. Immucube [45] significantly improves the over-saturation throughput of Immunet by demanding one VNet to use dimension-ordered routing. However, its high demand on VCs (at least three VNets for each message class) renders it more applicable for off-chip networks [45]. Vicis [19] uses the turn model for deadlock freedom and removes routing restrictions when the restrictions disconnect some nodes. However, deadlock freedom cannot be strictly guaranteed [2].

Researchers in ARIADNE [2] construct a spanning tree over the network to connect available links and routers, and utilize

[5]The idea of escape VC can be applied for either deadlock avoidance or deadlock recovery.

up*/down* [48] to avoid deadlocks after the network becomes faulty. uDIREC [40] extends the spanning tree construction to cover unidirectional link faults. An alternative approach for deadlock avoidance is segment-based routing [32], dividing a network into segments and placing different turn restrictions in each segment. BLINC [29] uses the basic idea of segment-based routing to reduce reconfiguration time, but it places requirements on the number and distribution of faults. However, the routing restrictions of all these approaches result in longer average hops and nonminimal routing, which further negatively affect both flit traversal latency and over-saturation throughput.

B. Deadlock Detection and Recovery Based Approaches

Deadlock recovery based approaches [10], [46], [47], [50] permit the existence of deadlocks but promise to recover from them. Nord [10] uses a unidirectional ring as the escape VC [14] in a power gated NoC but suffers from its long latency. Router Parking [50], instead, uses spanning tree to construct the escape VC. In Nord and Router Parking, the escape VC is enabled when a deadlock is detected according to a timeout mechanism or the total misrouted hops of a packet. Both of them have to reserve one VC for deadlock recovery even if there is no deadlock. Therefore, the saturation point is limited due to the wasted bandwidth. Besides, escape VC [14] based approaches place high demands on the number of VCs, and fail to achieve deadlock freedom and fully adaptive routing with one VC. Static Bubble [47] extends BFC from ring networks to achieve deadlock freedom on irregular topologies. SPIN [46] uses a similar distributed implementation of deadlock detection and recovery.

C. Subactive Deadlock Freedom Approaches

There is a series of approaches [36]–[39] that removes deadlock detection but develops method to recover from deadlocks. They claim themselves to be subactive [36], because by periodically forcing packet movement, deadlocks are naturally resolved. BBR [37] reserves bubble buffers in every router and uses the synchronized packet swapping between adjacent routers to handle deadlocks. BINDU [39] uses a fixed path to swap packets so as to reduce the required number of bubble buffers from one in every router in BBR to one in the whole network. SWAP [38] forces the progressive packets movement also via swapping and resolves deadlocks without any bubble buffers. DRAIN [36] uses periodical draining for deadlock recovery. These approaches share several characteristics. First, due to the lack of deadlock detection, they are late in resolving deadlocks that might have existed in the network for a long time. Second, they resolve deadlocks in a probabilistical manner, i.e. the forced packet movement may not address deadlocks but the probability for a deadlock to exist approaches zero as the number of recoveries increases. Therefore, their forced packet movement often incur many disroutes without effectively resolving deadlocks.

D. Bufferless Deflection

Bufferless deflection or backpressureless flow control based approaches [13], [15]–[18], [21], [22], [24], [26], [34], [49] are internally deadlock free, due to packets' constant routing and deflections. DeDR manages to apply the backpressureless flow control during the deflection mode for deadlock recovery. Compared with previous deflection approaches, DeDR offers the following changes. First, DeDR operates in the normal state with buffer backpressure and transitions to the backpressureless flow control only during the deadlock recovery. AFC [22] is the only existing deflection methods that have also proposed a transition between two flow controls. But AFC is proposed for higher energy efficiency when the traffic load is low. Its motivation and implementation are different to DeDR. Second, by combing the backpressureless flow control with a network with buffers, DeDR achieves a higher throughput and can further improve the performance by adding more VCs. Third, DeDR can handle protocol deadlocks. But previous deflection approaches cannot achieve protocol deadlock freedom because they cannot separate different message classes [33], [34]. SCEPTER [13] focus on router bypassing and source throttling on bufferless NoC. FastTrack [24] reduces the overhead when implementing bufferless NoC on FPGA. The Clumsy flow control [26] is a throttling mechanism on bufferless NoC to reduce the deflection rate. The Banyan network based switch is replaced by a Benes network in [49] to enable fault tolerance of allocation.

VII. Conclusion

Deadlocks and saturation are expected to be more frequent in NoCs with faulty links or routers. State-of-the-art deadlock-free approaches for faulty NoCs fail to sustain a high average throughput and satisfying fairness when the network becomes over-saturated. In this paper, we observe a positive feedback loop between deadlocks and congestions and propose a deadlock recovery framework that handles both deadlocks and congestions. A combination of probe detection mechanism and timeout counters is used for a low false positive rate and timely deadlock detection. For highly efficient deadlock recovery, a deflection mode is designed to use backpressureless flow control to eject all packets in the network. By transmitting special messages, the implementation of the deflection mode is carried out in a distributed manner and has a good scalability. DeDR reduces the average flit traversal latency and significantly improves the over-saturation performance. In future work, reducing the deflection rate and improving energy efficiency during the deflection mode will be considered.

References

[1] N. Agarwal et al. Garnet: A detailed on-chip network model inside a full-system simulator. In *ISPASS*, pages 33–42, 2009.

[2] K. Aisopos et al. Ariadne: Agnostic reconfiguration in a disconnected network environment. In *PACT*, pages 298–309, 2011.

[3] K. V. Anjan and T. M. Pinkston. An efficient, fully adaptive deadlock recovery scheme: Disha. In *ISCA*, pages 201–210, 1995.

[4] M. Badr and N. E. Jerger. Synfull: Synthetic traffic models capturing cache coherent behaviour. In *ISCA*, pages 109–120, 2014.

[5] P. Baran. On distributed communications networks. *IEEE Transactions on Communications Systems*, 12(1):1–9, 1964.

[6] E Baydal et al. Simple and efficient mechanism to prevent saturation in wormhole networks. In *IPDPS*, pages 617 – 622, 2000.

[7] Nathan Binkert et al. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, 2011.

[8] Kevin Chang et al. Hat: Heterogeneous adaptive throttling for on-chip networks. pages 9–18, 10 2012.

[9] L. Chen et al. Power punch: Towards non-blocking power-gating of noc routers. In *HPCA*, pages 378–389, 2015.

[10] L. Chen and T. M. Pinkston. Nord: Node-router decoupling for effective power-gating of on-chip routers. In *Micro*, pages 270–281, 2012.

[11] W. J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *DAC*, pages 684–689, 2001.

[12] William Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[13] B. K. Daya, L. Peh, and A. P. Chandrakasan. Quest for high-performance bufferless nocs with single-cycle express paths and self-learning throttling. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2016.

[14] J. Duato. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(10):1055–1067, 1995.

[15] C. Fallin et al. Chipper: A low-complexity bufferless deflection router. In *HPCA*, pages 144–155, 2011.

[16] C. Fallin et al. Minbd: Minimally-buffered deflection routing for energy-efficient interconnect. In *NOCS*, pages 1–10, 2012.

[17] Mohammad Fattah et al. A low-overhead, fully-distributed, guaranteed-delivery routing algorithm for faulty network-on-chips. In *NOCS*, pages 18:1–18:8, 2015.

[18] C. Feng et al. Addressing transient and permanent faults in noc with efficient fault-tolerant deflection router. *VLSI*, 21(6):1053–1066, 2013.

[19] David Fick et al. A highly resilient routing algorithm for fault-tolerant nocs. In *DATE*, pages 21–26, 2009.

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

[21] Mitchell Hayenga, Natalie Enright Jerger, and Mikko Lipasti. Scarab: A single cycle adaptive routing and bufferless network. In *MICRO*, MICRO 42, pages 244–254, 2009.

[22] S. A. R. Jafri et al. Adaptive flow control for robust performance and energy. In *Micro*, pages 433–444, 2010.

[23] Natalie Enright Jerger, Tushar Krishna, and Li-Shiuan Peh. On-chip networks, second edition. *Synthesis Lectures on Computer Architecture*, 12(3):1–210, 2017.

[24] N. Kapre and T. Krishna. Fasttrack: Leveraging heterogeneous fpga wires to design low-cost high-performance soft nocs. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 739–751, 2018.

[25] H. Kim, G. Kim, H. Yeo, J. Kim, and S. Maeng. Design and analysis of hybrid flow control for hierarchical ring network-on-chip. *IEEE Transactions on Computers*, 65(2):480–494, 2016.

[26] H. Kim, Y. Kim, and J. Kim. Clumsy flow control for high-throughput bufferless on-chip networks. *IEEE Computer Architecture Letters*, 12(2):47–50, 2013.

[27] S. Konstantinidou and L. Snyder. The chaos router: A practical application of randomization in network routing. *SIGARCH Comput. Archit. News*, 19(1):79–88, 1991.

[28] T. Krishna, C. O. Chen, W. C. Kwon, and L. Peh. Breaking the on-chip latency barrier using smart. In *HPCA*, pages 378–389, 2013.

[29] D. Lee et al. Brisk and limited-impact noc routing reconfiguration. In *DATE*, pages 1–6, 2014.

[30] Z. Li et al. The runahead network-on-chip. In *HPCA*, pages 333–344, 2016.

[31] Sheng Ma et al. Dbar: An efficient routing algorithm to support multiple concurrent applications in networks-on-chip. In *ISCA*, pages 413–424, 2011.

[32] A. Mejia et al. Segment-based routing: an efficient fault-tolerant routing algorithm for meshes and tori. In *IPDPS*, pages 10 pp.–, 2006.

[33] G. Michelogiannakis, D. Sanchez, W. J. Dally, and C. Kozyrakis. Evaluating bufferless flow control for on-chip networks. In *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, pages 9–16, 2010.

[34] Thomas Moscibroda and Onur Mutlu. A case for bufferless routing in on-chip networks. In *ISCA*, pages 196–207, 2009.

[35] George Nychis et al. On-chip networks from a networking perspective: Congestion and scalability in many-core interconnects. *ACM SIGCOMM Computer Communication Review*, 2012.

[36] M. Parasar, H. Farrokhbakht, N. Enright Jerger, P. V. Gratz, T. Krishna, and J. San Miguel. Drain: Deadlock removal for arbitrary irregular networks. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 447–460, 2020.

[37] M. Parasar, A. Sinha, and T. Krishna. Brownian bubble router: Enabling deadlock freedom via guaranteed forward progress. In *NOCS*, pages 1–8, 2018.

[38] Mayank Parasar, Natalie Enright Jerger, Paul V. Gratz, Joshua San Miguel, and Tushar Krishna. Swap: Synchronized weaving of adjacent packets for network deadlock resolution. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, page 873–885, 2019.

[39] Mayank Parasar and Tushar Krishna. Bindu: Deadlock-freedom with one bubble in the network. In *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, 2019.

[40] R. Parikh and V. Bertacco. udirec: Unified diagnosis and reconfiguration for frugal bypass of noc faults. In *MICRO*, pages 148–159, 2013.

[41] S. Park et al. Approaching the theoretical limits of a mesh noc with a 16-node chip prototype in 45nm soi. In *DAC*, pages 398–405, 2012.

[42] G. F. Pfister and V. A. Norton. "hot spot" contention and combining in multistage interconnection networks. *IEEE Transactions on Computers*, C-34(10):943–948, 1985.

[43] V. Puente, R. Beivide, J. A. Gregorio, J. M. Prellezo, J. Duato, and C. Izu. Adaptive bubble router: a design to improve performance in torus networks. In *ICPP*, pages 58–67, 1999.

[44] V. Puente et al. Immunet: a cheap and robust fault-tolerant packet routing mechanism. In *ISCA*, pages 198–209, 2004.

[45] V. Puente and J. A. Gregorio. Immucube: Scalable fault-tolerant routing for k-ary n-cube networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(6):776–788, 2007.

[46] A. Ramrakhyani et al. Synchronized progress in interconnection networks (spin): A new theory for deadlock freedom. In *ISCA*, pages 699–711, 2018.

[47] A. Ramrakhyani and T. Krishna. Static bubble: A framework for deadlock-free irregular on-chip topologies. In *HPCA*, pages 253–264, 2017.

[48] Thomas L. Rodeheffer et al. Autonet: A high-speed, self-configuring local area network using point-to-point links. *IEEE Journal on Select Areas of Communication*, 9, 1991.

[49] A. Runge and R. Kolla. Using benes networks at fault-tolerant and deflection routing based network-on-chips. In *NOCS*, pages 1–8, 2016.

[50] A. Samih et al. Energy-efficient interconnect via router parking. In *HPCA*, pages 508–519, 2013.

[51] Daniel J. Sorin, Mark D. Hill, and David A. Wood. A primer on memory consistency and cache coherence. *Synthesis Lectures on Computer Architecture*, 6(3):1–212, 2011.

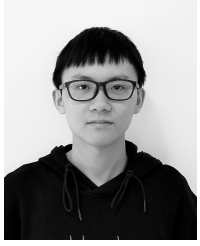
[52] C. Sun et al. Dsent - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In *NOCS*, pages 201–210, 2012.

[53] Mithuna Thottethodi et al. Self-tuned congestion control for multiprocessor networks. pages 107 – 118, 2001.

[54] Hassan M. G. Wassel, Ying Gao, Jason K. Oberg, Ted Huffmire, Ryan Kastner, Frederic T. Chong, and Timothy Sherwood. Surfnoc: A low latency and provably non-interfering approach to secure networks-on-chip. In *ISCA*, page 583–594, 2013.

[55] Yong Ho Song and T. M. Pinkston. A new mechanism for congestion and deadlock resolution. In *ICPP*, pages 81–90, 2002.

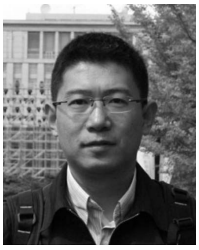
[56] Yong Ho Song and T. M. Pinkston. A progressive approach to handling message-dependent deadlock in parallel computer systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3):259–275, 2003.



Yibo Wu received the BEng degree in microelectronics from Xidian University, China, in 2018. He is currently persuing the MSc degree at Institute of Microelectronics, Tsinghua University, China. His research interests include reliability-aware design for network-on-chip and many-core system.

Liang Wang received the BEng and MSc degree in electronics engineering from Harbin Institute of Technology, China, in 2011 and 2013 respectively, and the Ph.D degree in Computer Science and Engineering from The Chinese University of Hong Kong, Hong Kong, in 2017. He is currently an assistant professor with the School of Computer Science and Engineering, Beihang University, China. Previously, he was a postdoctoral research fellow in Institute of Microelectronics, Tsinghua University. His research interests include power-efficient and reliability-aware design for network-on-chip and many-core system.

Xiaohang Wang received the BEng and PhD degrees in communication and electronic engineering from Zhejiang University, Hangzhou, China, in 2006 and 2011, respectively. He is currently an Associate Professor with the South China University of Technology, Guangzhou, China. His current research interests include manycore architecture, power efficient architectures, optimal control, and network-on-chip-based systems. Dr. Wang was a recipient of the PDP 2015 and VLSI-SoC 2014 Best Paper Awards.



Jie Han received the BSc degree in electronic engineering from Tsinghua University, Beijing, China, in 1999, and the PhD degree from Delft University of Technology, Delft, The Netherlands, in 2004. He is currently a Professor in the Department of Electrical and Computer Engineering, the University of Alberta, Edmonton, AB, Canada. His research interests include approximate computing, stochastic computation, reliability and fault tolerance, nanoelectronic circuits and systems, novel computational models for nanoscale and biological applications. Dr. Han and coauthors received the Best Paper Award at the International Symposium on Nanoscale Architectures 2015 (NanoArch 2015) and Best Paper Nominations at the 25th Great Lakes Symposium on VLSI 2015 (GLSVLSI 2015), NanoArch 2016 and the 19th International Symposium on Quality Electronic Design (ISQED 2018).

Shouyi Yin received the B.S., M.S., and Ph.D. degrees in electronic engineering from Tsinghua University, China, in 2000, 2002, and 2005, respectively. He was with the Imperial College London as a Research Associate. He is currently a Professor with the Institute of Microelectronics, Tsinghua University. He has authored or co-authored over 40 refereed papers, and served as TPC member or reviewers for the international key conferences and leading journals. His research interests include SoC design, reconfigurable computing, and mobile computing.

Shaojun Wei was born in Beijing, China, in 1958. He received the Ph.D. degree from the Faulté Polytechnique de Mons, Belgium, in 1991. He is currently a Professor with the Institute of Microelectronics, Tsinghua University. He is also a Senior Member of the Chinese Institute of Electronics. His main research interests include VLSI SoC design, EDA methodology, and ASIC design.

Leibo Liu received the BS degree in electronic engineering from the Tsinghua University, Beijing, China, in 1999 and the PhD degree in the Institute of Microelectronics, Tsinghua University, in 2004. He now serves as a Tenured Professor in the Institute of Microelectronics, Tsinghua University. His research interests include reconfigurable computing, mobile computing and VLSI DSP.