Bufferless NoCs with Scheduled Deflection Routing

Chen Chen, Zirui Tao, Joshua San Miguel
Department of Electrical and Computer Engineering
University of Wisconsin–Madison, WI, USA
Email: {cchen532, ztao23, jsanmiguel}@wisc.edu

***Abstract*—Bufferless networks-on-chip (NoCs) with deflection routing are a promising approach for saving area and power in the communication fabric of many-core processors. However, deflection routing imposes fundamental limitations on the number of input and outports per router, which compromises the NoCs' ability to adapt to permanent link failures due to wearout. The susceptibility to wearout of modern NoCs has increased due to deeper sub-micro CMOS process technology. In this paper, we propose *scheduled deflections for bufferless NoCs* (SchedNoC), a framework that prolongs the lifetime of bufferless NoCs via strategic time-multiplexing of links, which allows links to fail arbitrarily while ensuring correct network operation. We show that SchedNoC can increase the NoCs' lifetime by 62.5%–71.8% at only 4%–15% performance degradation compared to conventional deflection routing.**

I. Introduction

Modern packet-switched networks-on-chip (NoCs) require expensive buffering to provide high-performance, high-bandwidth packet transactions. However, these buffers incur substantial hardware overhead [1] and compromise the area and power efficiency of today's chip multi-processors (CMPs).

Promising Approach: Bufferless NoCs. *Bufferless NoCs* (e.g., BLESS [2]) have shown to be a promising solution for scalable interconnect area and power. The idea is to keep packets in constant motion and remove the buffers in the routers. In this way, bufferless NoCs avoid storing packets by forcing them to leave the router immediately after deciding which way they need to go. However, multiple incoming packets may vie for the same outport of a router during route computation. Bufferless NoCs apply the principle of *deflection routing* to pick one winner packet and deflect others to different non-preferred outports, potentially misrouting them. Prior work on BLESS [2] demonstrates 40% energy reduction while maintaining high performance for applications with moderate network utilization. Thus bufferless NoCs with deflection routing can prove to be a scalable low-cost solution for future many-core processors.

Challenge 1: Deflection Routing Constraints Unfortunately, deflection routing imposes strict design requirements, namely *port number constraint*, on the network topology in order to operate correctly. It requires that *every router have at least as many outports available for all of its incoming packets*. In other words, the number of inports must never exceed the number of outports in any router. This restriction makes bufferless NoCs difficult to use for irregular topology. But more importantly, as in the focus of this paper, this limitation compromises the bufferless NoCs' ability to adapt to arbitrary link failures due to wearout in electronic components, degrading the effective lifetime of NoCs (and potentially the lifetime of the entire CMP).

Challenge 2: Wearout and Lifetime of NoCs Today's CMPs have become more vulnerable to wearout due to higher transistor density and deeper CMOS process technology. From the perspective of the interconnect, this implies that *permanent link failures* are becoming non-trivial and far more frequent [3]. These link failures do not necessarily immediately compromise the functionality of the CMP, because systems can usually accommodate redundant cores. But the more permanent link failures that occur, the higher the likelihood of causing NoCs to be *disconnected*, thus making some cores, memory and I/O devices inaccessible eventually [4]. We define that a NoC reaches the end of its *lifetime* when there exists at least one router that can neither send to nor receive packets from at least one other router. As will be discussed in detail in Section II, prior strategies for tolerating faulty links are often too conservative, particularly with bufferless NoCs, and thus struggle to extend the lifetime of systems.

Solution: Scheduled Deflections. In this paper, we propose *Scheduled Deflections for Networks-on-Chip (SchedNoC)*, a bufferless NoC architecture that employs *scheduled deflection routing* to enable continuous, correct execution of the bufferless NoC even in the presence of permanent link failures due to wearout. SchedNoC enables a *Unidirectional Link Disconnection (ULD)* strategy that overcomes the port number constraint (Challenge 1) by dynamically multiplexing packets via *time slices* so that the number of outports can safely be less than the number of inports in any router and all incoming packets can always find available outports to traverse. This subsequently addresses Challenge 2 because we can now allow links to fail arbitrarily with much less concern of the correct operation of the bufferless NoC.

Contributions. Our work contributes the following:

- Introduces the concept of *scheduled deflections* for bufferless NoCs to greatly improve resilience to wearout.
- Proposes *SchedNoC*, a bufferless NoC architecture that allocates time slices on non-faulty links to avoid conflicts when the number of outports is not sufficient given the number of inports. This makes the interconnect robust to permanent link failures and makes bufferless NoCs a viable option for irregular topologies.
- Presents two allocation algorithms for deriving time slices, *Multi-Ring* and *Even-Odd*, which allow SchedNoC to maximize network lifetime. Our evaluation of SchedNoC with our Multi-Ring algorithm demonstrates 71.8% increased lifetime at modest 15% performance cost compared to conventional bufferless NoCs with 10-30%

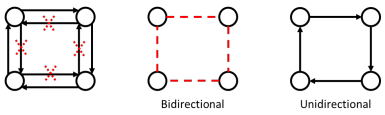


Fig. 1. 2x2 Mesh with 4 faults, comparing BLD and ULD

area and power overhead. With Even-Odd, SchedNoC increases lifetime by 62.5% with very low performance degradation (4%) and overhead (less than 1%).

II. Motivation

Link Wearout and NoC Lifetime. The lifetime and resiliency to wearout of NoCs are increasingly important concerns. Though prior works [3], [5] provide mechanisms for handling faulty links in NoCs, they are fundamentally too conservative and do not consider the side-effect on lifetime, employing what we refer to as a *bidirectional link disconnection* (BLD) strategy. In BLD mechanisms, when a link fails, the system disconnects both the faulty unidirectional link as well as the unidirectional link going in the opposing direction even though the opposing link may not be faulty. This is often meant to keep the hardware simple, because an imbalance in the direction of links makes routing and flow control non-trivial. More importantly, as explained in Section I, BLD is necessary for bufferless NoCs so as to not violate port number constraints. Though the BLD strategy ensures correctness, it is fundamentally too conservative and can quickly degrade the lifetime of a NoC.

Ideally, we would prefer a *unidirectional link disconnection* (ULD) strategy instead, where only the faulty links are disconnected. This maximizes NoC's lifetime to its theoretical limit. A brief example is shown in Fig. 1, where employing BLD for failed links results in an unusable NoC, and thus the NoC reaches the end of its lifetime, because some routers are no longer able to send or receive packets from some other routers; i.e., the network becomes disconnected. On the other hand, employing ULD is able to retain network connectivity despite many faulty links.

Degradation in NoC Lifetime due to BLD. We characterize the amount of NoC lifetime wasted due to BLD. Specifically, we show how many links must fail (on average) before a NoC reaches the end of its lifetime, comparing BLD and ULD strategies. Fig. 2 shows the results of our characterization for varying sizes of 2D meshes. For each topology, we iteratively increase the number of faulty links, disconnecting links in a uniform random fashion. We run 1000 experiments per iteration. The figure shows how many links needed to fail such that in 90% of the 1000 experiments, the NoC reached the end of its lifetime (i.e., became disconnected and unusable). For example, using BLD, more than 900 experiments with 53 faulty links on the 10×10 mesh lead to unusable NoCs, whereas with ULD, the same topology can withstand up to 88 faulty links, which implies 66% longer lifetime when using ULD. On average, across the varying topologies, we find

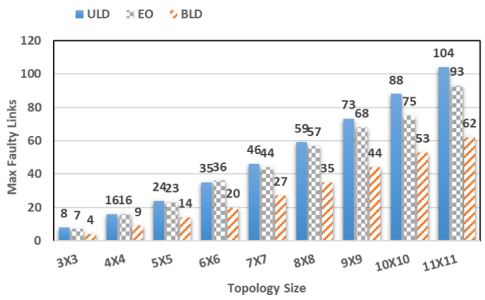


Fig. 2. Number of links that must fail such that the NoC reaches the end of its lifetime

that employing the novel ULD approach prolongs lifetime by 71.8% on average compared to conventional BLD.

III. SchedNoC Concept

We introduce the *Scheduled Deflection Network-on-Chip* (SchedNoC), a framework to build resilient bufferless NoC. SchedNoC enables ULD for handling wearout in bufferless NoCs, overcoming their port number constraints, and significantly prolonging network lifetime.

The key concept is that SchedNoC allocates *time slices* on each link, effectively time-multiplexing the possible turns between links. Each link can only transfer packets when the clock hits its preassigned time slice period. Consider the example in Fig. 3a. This bufferless router has three input ports and two output ports, which violates the port number constraint. In order to operate correctly, a conventional bufferless NoC would need to drop packets, which would be expensive to support, requiring control networks for acknowledgments and additional buffers at the endpoints (thus making the NoC no longer bufferless). However, through careful time slice allocation on each link, SchedNoC is able to overcome this constraint. In Fig. 3b and 3c, SchedNoC allocates time slice "0" and "1" on two different pairs of input ports. When the clock hits time slice "0", only the pair of ports allocated with "0" can transfer packets, and similarly for time slice "1". Thus, at each clock cycle, no more than two incoming packets can arrive at this router simultaneously, and they can always find available output ports to turn towards. *Takeaway: SchedNoC guarantees that the number of available input ports is dynamically not greater than the number of available output ports at any clock cycle, even though the presence of faulty links introduces the violation of the port number constraint.*

The time-multiplexed allocation of links is global. We assume that all routers in a NoC share the same global clock, which is the common case in modern packet-switched NoCs. The global clock is broken down into short periods of cycles called *system time slices* (syslices). The ports with allocated time slices equal to the current value of the syslice will be time-multiplexed as available by routers. The range of syslices is defined as the total number of distinct time slices (TTS) across the entire NoC. In Fig. 3, there are three different time slices: "0,1,2". Therefore, the TTS is three in this case, and

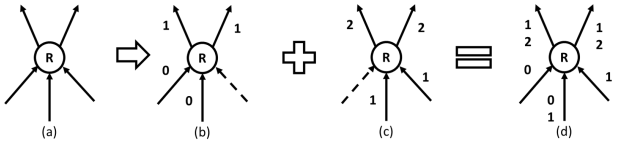


Fig. 3. SchedNoC concept of allocating time slices on a router with three input ports and two output ports

the clock will be split into three periods. Thus the syslice will update in round-robin order as "0,1,2,0,1,2..." throughout the lifetime of the NoC.

The cycle duration between syslices is determined by the per-hop latency, including link traversal delay. If the per-hop latency is assumed to be two clock cycles (e.g., one cycle in router pipeline and one cycle traversing the link), the syslice will update every two clock cycles.

IV. SchedNoC Algorithm Concept

A. Algorithm Design Goals

The goal of our SchedNoC algorithm is to allocate time slices on active (non-faulty) links to ensure correct bufferless operation. An ideal allocation algorithm should maximize the *lifetime* of the NoC and minimize *area overhead*, *power overhead* and *performance degradation*.

Lifetime. We assume that the upperbound of a NoC lifetime is equivalent to what its lifetime would be had it strictly employed the ULD strategy (i.e., only faulty links are ever disconnected). Technically, an allocation algorithm is allowed to *logically disconnect extra non-faulty links* upon a fault, similar to BLD. Disabling non-faulty links can be seen as simply not allocating any time slices for these links, so that their availability will not be time-multiplexed. The Even-Odd algorithm that we introduce in Section V is an example approach that requires logically disconnecting additional non-faulty links.

Area and Power. SchedNoC routers require tiny buffers to store time slices on each non-faulty link. Intuitively, the main area and power overhead are from these buffers. In order to reduce buffer size, allocation algorithms need to minimize TTS, which is the number of distinct time slices required to guarantee correct system operation because in our SchedNoC router microarchitecture, time slice allocations are stored in a bitvector, whose size is proportional to TTS. A more detailed microarchitecture description is provided in Section VI, and the area and power evaluations are provided in Section VII.

Performance. To minimize the performance degradation compared to conventional bufferless NoCs, allocation algorithms need to maximize the average throughput. Because SchedNoC dynamically time-multiplexes the availability of links, the throughput of each link can be seen as *the number of time slices allocated on it divided by TTS*. For example, in Fig. 3, the TTS is three; Because the right input port is allocated only one time slice, this means that every three cycles, this link can only transfer one packet. Therefore, the throughput of this link is 33%. The upperbound of throughput is 1, which means a link is allocated with all time slices and can transfer packets at any time. An algorithm that employs BLD can generally achieve the upperbound, because it guarantees that the number of available input ports is always equal to the number of available output ports. A SchedNoC algorithm generally trades off throughput with area and power because with a small TTS, deallocating one time slice from a link has more significant impact on throughput.

B. Algorithm Design Principles

The effect of time-multiplexing each link propagates throughout the NoC topology. Thus, the SchedNoC allocation algorithm can be expressed as a graph problem. We assume that a network in its lifetime is one where all routers are able to send and receive packets from all other routers. If at least one router becomes disconnected and can no longer communicate with at least one other router, then the NoC is unusable and has reached the end of its lifetime. In this paper, the algorithms are developed for the NoC that is still in its lifetime. We follow two key principles in designing our SchedNoC algorithms:

- The total number of input ports in a router with the same time slice must be no greater than the total number of output ports with the same successive time slice.
- Time slices should be allocated at the granularity of rings of non-faulty links.

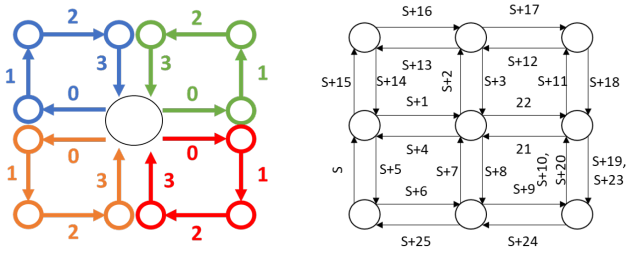
1) Time slices constraint: SchedNoC overcomes the *port number constraint* by dynamically time-multiplexing the availability of links, allowing the number of input ports to *physically* exceed the number of output ports in a router. However, to do so, at each cycle, SchedNoC still needs to ensure that the number of *available* input ports does not exceed the number of *available* output ports, so that incoming packets can always find available output ports on which to exit. This requirement introduces a new constraint of SchedNoC, called *time slices constraint*, that means *the total number of input ports with the same time slice must be less than or equal to the total number of output ports with the same successive time slice.* The successive time slice on output ports means the time slice at the input ports *plus 1*. Note that when the time slice value accumulates up to TTS, it resets back to 0. To simplify our implementations, we design our algorithms such that the number of input ports with the same time slice is exactly **equal to** the number of output ports with the same successive time slice.

2) Follow rings to allocate time slices: Because bufferless NoCs employ deflection routing, packets must leave each router right after finishing the transaction process at the router, and keep hopping among routers until they arrive at their destination. We recognize that for any connected network, the set of all non-faulty links can be decomposed into multiple rings, because any router can send to or receive packets from all other routers. SchedNoC can follow *decomposed rings* to allocate time slices on links. Within a ring, the number of input ports always equals the number of output ports, which obeys the time slice constraint; thus allocating time slices along rings

can guarantee any incoming packets can always find available output ports at any router.

Each decomposed ring is not required to cover all routers nor all non-faulty links in the network, but the final combination of rings must do so. Multiple rings may contain the same routers, which we refer to as *interaction routers*. Because the destination of a packet may not be located at a router within the current ring, the packet must eventually switch to another ring via these interaction routers. A packet may need to hop along multiple rings to get to its destination.

To simplify design complexity, we impose the rule that all decomposed rings have a size (in links) that is proportional to the TTS of the network. For example, a size-2 ring and a size-4 ring can both accommodate a TTS of 2. This makes it simpler for packets to traverse through interaction routers and switch rings when needed.



(a) Time slices allocation following four rings

(b) The first size-26 ring

Fig. 4. Ring allocation strategy

Consider the example topology in Fig. 4a. It can be decomposed into four rings each of size 4, with the TTS set to 4. Allocated time slices following them guarantees the number of available output ports at each router are always equal to the number of available input ports at any time. Packets can change ring tracks at interaction routers to arrive at different destinations. In this example, there is only one interaction router in the middle. All packets can arrive at this interaction router when *syslice* is in period "3" and change their ring track when syslice is in period "0".

V. Proposed SchedNoC Algorithms

In this section, we introduce our proposed algorithms for allocating time slices to links in SchedNoC. Whenever a link fails due to wearout, the first step is to verify that the NoC has not reached the end of its lifetime (i.e., the network topology is still connected). This verification can be achieved via a breath-first or depth-first traversal across all routers. Upon a newly faulty link, SchedNoC invokes its allocation algorithm to reassign time slices to all non-faulty links. We propose two allocation algorithms, *Multi-Ring* and *Even-Odd* and analyze them in terms of throughput, TTS and lifetime in Section VII.

Multi-Ring (MR) Algorithm. Our MR algorithm follows directly from our algorithm design principles discussed in Section IV-B. The MR algorithm aims to apply ULD directly and utilizes all non-faulty links. It is designed for all kinds

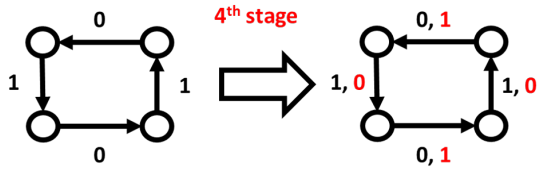


Fig. 5. Allocating more time slices at the fourth MR stage

of topologies and is divided into four stages. The paths of a NoC can be decomposed into multiple rings of the same size. The **first stage** is to generate one ring with a specific size, by randomly selecting non-faulty links. To simplify the algorithm design, we force the *first-generated ring to cover all non-faulty links*. Fig. 4b shows a first-generated ring on a standard 3x3 mesh. The "S" represents the first selected non-faulty link and "+number" represents the order that other non-faulty links join the ring. Note that some non-faulty links may be covered multiple times; e.g., the link allocated with "S + 10" and "S + 20". In this case, the size of the ring is 26, and in the MR algorithm, we set the TTS of a NoC equal to the size of this ring.

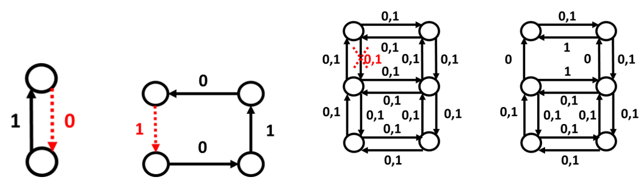
The **second stage** is to generate a group of rings of the same size. These rings must be distinct; their tracks must not be identical. The **third stage** is to follow rings in this group one-by-one to allocate time slices. If the new time slices obey the time slices constraint (Section IV-B), they are retained, and the ring is deemed eligible. Our algorithm will then follow the next ring in the group to allocate time slices until all rings in the group are processed. To maximize the average throughput of non-faulty links, we employ a **fourth stage** to allocate even more time slices on each eligible ring. An example irregular 2x2 mesh is shown in Fig. 5, which can only be decomposed into one ring. Assuming the TTS of the NoC is 2, the third stage allocates time slices one round following the ring. At the fourth stage, the algorithm finds that allocating time slices one more round following the ring will not violate the time slices constraint and can improve throughput.

Even-Odd (EO) Algorithm. Unlike MR, our EO algorithm initially allocates links with all possible time slices and gradually removes time slices on faulty links. The EO algorithm applies ULD but requires *logically disconnecting extra non-faulty links*. Furthermore, the EO algorithm is only designed for mesh-based topologies; extending to others is left for future work. In meshes, we observe that the time slices only need to be either even or odd. Thus, to simplify the algorithm design, we define the TTS to 2, yielding only time slices "0" and "1". The allocation process can be divided into three stages.

The **first stage** is to convert the current topology of a NoC to a *regular topology*, so that the algorithm can allocate all possible time slices on the links. We define a regular topology to be one where *the number of inports and the number of outports are equal at all routers*, so that each unidirectional link can only have one another unidirectional link with the opposing direction. A fully-connected 2D mesh is an example

of a regular topology. By this definition, incoming packets can always find available outports to transfer, so that the NoC can keep links available, and in the case of our EO algorithm, we can allocate all possible time slices on the links.

The **second stage** is to remove time slices on faulty links, which includes links that we manually added to make the topology regular. This removal process needs to abide by the time slices constraint (Section IV-B). In order to remove a time slice from a faulty link, the algorithm needs to traverse along some ring, removing a pair of time slices at each router, one from an inport and another with the successive slice from an outport. Because we want to minimize the throughput loss due to removing time slices, we aim to find the smallest ring that can be used to remove time slices on a faulty link. For mesh-based NoCs, the smallest ring is a size-2 ring, which is a pair of links with opposing directions, and the second smallest ring is a size-4 ring that connects four routers in a rectangle. Fig. 6a and Fig. 6b give an example of how a faulty link (dashed line) follows the two smallest rings to remove time slices on it. Fig. 6c shows a converted regular topology with one faulty link. Fig. 6d shows the result of our EO algorithm after removing time slices on this faulty link.



(a) (b) (c) (d)

Fig. 6. An example of removing time slices on a faulty link

In the process of removing time slices, the EO algorithm may logically disconnect additional non-faulty links. A non-faulty link can be traversed by rings that belong to multiple faulty links. Once all of the non-faulty link's time slices are removed, the link becomes logically disconnected. Thus the **last stage** of our algorithm is to verify that the NoC is still within its lifetime (i.e., still connected and usable). This can be simply achieved by iterating through each router and checking to see if it can still send to and receive packets from all other routers. Note that the logical disconnection is not permanent because the link has not physically failed. If a new faulty link appears, and the NoC must rerun the EO algorithm, these logically disconnected non-faulty links are reintroduced.

VI. Proposed SchedNoC Architecture

SchedNoC's architecture follows from conventional bufferless NoCs and introduces very little overhead to the router microarchitecture. The main difference is when arbitrating for outports, the allocators now need to check the outports' availability based on the current syslice. In general, a NoC employs a routing table to find the outport with the appropriate path to the destination during route computation. As in prior approaches for handling faults, when a new faulty link is detected, the routing table must be correspondingly updated [5].

TABLE I

System parameters used in our simulations

Multiprocessor	4x4 ALPHA cores
Coherence Protocol	MESI
Private L1 Cache	32KB per core, 4-way
Shared L2 Cache	128KB per core, 8-way
Interconnection Links	1-cycle, 128-bit flits
Bufferless Router	1-cycle pipeline
Frequency	1GHz

Checking Time Slices. SchedNoC requires a table of allocated time slices per outport as well as control signals for asserting the availability of outports. The time-multiplexing process of links' availability can be out of the critical path, because when there is an incoming packet, the route computation process requires one more cycle to decode the packet's destination. Therefore the update process can finish execution one cycle faster than route computation and pass signals to the crossbar and the allocator.

Storing Time Slices. Storing the exact values of time slices may incur high area overhead. Instead, we opt to encode the time slices via a bitvector per outport. The time slice value indexes into the bitvector, and the vector size is equal to the TTS. To check the availability of a link, the current syslice value is used to index into the bitvector. If the indexed bit is 1, the link is deemed available.

VII. Evaluation

We evaluate NoC lifetime, performance (synthetic and application) and hardware cost (network area and power), using parameters shown in Table I.

Lifetime. SchedNoC with our MR algorithm is able to utilize all non-faulty links, extending the NoC lifetime by the maximum amount (71.8%) over the conventional BLD approach. The comparison between MR (i.e., ULD), EO and BLD is shown in Fig. 2. Our EO algorithm has less (though still significant) lifetime improvement of 62.5% because it necessitates logically disconnecting some non-faulty links.

Performance. We first compare our two proposed SchedNoC algorithms analytically. Because SchedNoC dynamically time-multiplexes the availability of links, we use the *allocated rate* between the number of allocated time slices on each link (TS) and the total number of time slices (TTS) to evaluate the throughput of a link, computed as: $\frac{TS}{TTS}$. We can estimate the average throughput of a NoC by calculating the average allocated rate across all links. A large TTS requires routers to consume more power and space to store time slices. Therefore, a good allocation algorithm should maximize the average throughput while minimizing the TTS. We use this criterion to evaluate our proposed MR and EO algorithms. We use a 4x4 irregular mesh with 4, 8 and 12 arbitrary faults as test cases. We run each algorithm on each test case 1000 times to calculate the average throughput. The average throughput of our MR algorithm is found to be around 19%, 19% and 25% for the three test cases, respectively, while the results of our EO algorithm are 82%, 63% and 89%, respectively.

TABLE II
RESULTS OF EMPIRICAL EVALUATION

Algorithm	Lifetime	Performance Degradation	Area & Power
MR	71.8%	15%	10%
EO	62.5%	4%	<1%

Though currently only applicable to mesh-based topologies, we find that our EO algorithm achieves higher throughput. We observe that the TTS of a NoC with MR is highly related to the number of non-faulty links in the NoC because the randomly generated ring must cover all non-faulty links at least one time. Thus MR's scalability is limited by increasing area and power demands by increasing TTS. However, the TTS of a NoC with the EO algorithm is consistently "2" regardless of the NoC size. Therefore, the EO algorithm has better scalability and lower performance degradation than the MR algorithm, which is a worthy trade-off for its limited applicability to mesh-based topologies.

For our empirical evaluations, we use a standard bufferless NoC, BLESS [2], as our baseline. As test cases, we use 4x4 irregular meshes with 4, 8 and 12 arbitrary faults. The results of our BLESS baseline are from simulation on a regular 4x4 mesh with no faults. We first measure average packet latency as functions of injection rate for synthetic uniform random traffic. The results show that the non-faulty baseline saturates at 0.20 (packets/cycle/core), while SchedNoC with MR saturates at 0.06 (p/c/c), and SchedNoC with EO saturates at 0.18 (p/c/c). For application-level performance, we run full-system simulations of PARSEC benchmarks [6], [7] on gem5 [8] and Garnet2.0 [9]. Compared to the baseline, SchedNoC with MR maintains only 15% average performance degradation on our benchmarks while SchedNoC with EO maintains only 4% performance degradation.

Area and Power. We use DSENT [10] to analyze area and power consumption. We set Tri-Gate (Multi-Gate) 11nm LVT process as our electrical technology model. We use regular 4x4, 6x6 and 8x8 mesh as test cases and calculate average TTS, respectively. Compared to the baseline, the overhead of SchedNoC (both area and power) with MR is 10%, 18% and 30%, respectively. We find that the overhead with EO is negligible (less than 1%) regardless of the mesh size. The lifetime extension of each algorithm, the average normalized performance degradation results and area & power overhead results of 4x4 mesh are shown in Table II.

VIII. RELATED WORK

Prior efforts aim to prolong CMP lifetime proactively. The relationship between interconnect wearout and router's workload has been analyzed by [4]. They implement architectures to keep workloads at moderate traffic levels. [11] observes that reducing the variance of activity can avoid early failures. NBTI and HCI are two main causes of wearout [12]. Prior works on fault-tolerant NoCs are often too conservative on dynamic faulty links [3], [5], [13] and apply the BLD approach. [14] implements extra hardware to surround a NoC to enable fault tolerance, which has low scalability at larger NoC sizes.

IX. CONCLUSION

In this paper, we propose *scheduled deflections for bufferless NoCs (SchedNoC)*, a framework that prolongs the lifetime of bufferless NoCs. We introduce two novel algorithms, Multi-Ring (MR) and Even-Odd (EO). Both algorithms are designed for scheduling packet deflections and allow links to fail arbitrarily while ensuring correct network execution. We show that SchedNoC can increase the NoC lifetime by 62–71.8% at only 4–15% performance degradation compared to conventional deflection routing.

REFERENCES

[1] Paul Gratz, Changkyu Kim, Robert Mcdonald, Stephen W. Keckler, Doug Burger, "Implementation and evaluation of on-chip network architectures," in *2006 International Conference on Computer Design*, 2006.

[2] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3, pp. 196–207, 2009.

[3] Z. Zhang, A. Greiner, and S. Taktak, "A reconfigurable routing algorithm for a fault-tolerant 2d-mesh network-on-chip," in *2008 45th ACM/IEEE Design Automation Conference*, pp. 441–446, IEEE, 2008.

[4] H. Kim, A. Vitkovskiy, P. V. Gratz, and V. Soteriou, "Use it or lose it: Wear-out and lifetime in future chip multiprocessors," in *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 136–147, IEEE, 2013.

[5] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant nocs," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 21–26, European Design and Automation Association, 2009.

[6] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pp. 72–81, 2008.

[7] M. Gebhart, J. Hestness, E. Fatehi, P. Gratz, and S. W. Keckler, "Running parsec 2.1 on m5," *The University of Texas at Austin, Department of Computer Science, Tech. Rep*, 2009.

[8] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

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

[10] C. Sun, C.-H. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "Dsent-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, pp. 201–210, IEEE, 2012.

[11] W. Song, S. Mukhopadhyay, and S. Yalamanchili, "Architectural reliability: Lifetime reliability characterization and management ofmany-core processors," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 103–106, 2014.

[12] F. Oboril and M. B. Tahoori, "Extratime: Modeling and analysis of wearout due to transistor aging at microarchitecture-level," in *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, pp. 1–12, IEEE, 2012.

[13] M. Ebrahimi, M. Daneshtalab, J. Plosila, and F. Mehdipour, "Md: minimal path-based fault-tolerant routing in on-chip networks," in *2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 35–40, IEEE, 2013.

[14] Y. Kurokawa and M. Fukushi, "Design of an extended 2d mesh network-on-chip and development of a fault-tolerant routing method," *IET Computers & Digital Techniques*, vol. 13, no. 3, pp. 224–232, 2019.