Towards Maximum Utilization of Remained Bandwidth in Defected NoC Links

Chen, Changlin; Fu, Yaowen; Cotofana, Sorin

DOI
10.1109/TCAD.2016.2570680

Publication date
2017

Document Version
Accepted author manuscript

Published in
IEEE Transactions on Computer - Aided Design of Integrated Circuits and Systems

Towards Maximum Utilization of Remained Bandwidth in Defected NoC Links

Changlin Chen, *Student Member, IEEE,* Yaowen Fu, *Member, IEEE,* and Sorin Cotofana, *Senior Member, IEEE*

***Abstract*—To maximize the utilization of the available Networks-on-Chip (NoC) link bandwidth, partially faulty links with low fault level should be utilized while Heavily Defected (HD) links should be deactivated and dealt with by means of a fault tolerant routing algorithm. To reach this target, we make the following contributions in this paper: (i) We propose a Flit Serialization (FS) method to efficiently utilize partially faulty links. The FS approach divides the links into a number of equal width sections, and serializes sections of adjacent flits to transmit them on all fault-free link sections to mitigate the unbalance between the flit size and the actual link bandwidth. (ii) We propose the link augmentation with one redundant section as a low cost mechanism to mitigate the FS drawback that a link's available bandwidth is reduced even if it contains only 1 faulty wire. (iii) We deactivate HD links when their fault level exceed a certain threshold to diminish congestion caused by HD links. The optimal threshold is derived by comparing the zero load packet transmission latency on the HD links and that on the shortest alternative path. Our proposal is evaluated with synthetic traffic and PARSEC benchmarks. Experimental results indicate that the FS method can achieve lower area*power/saturation_throughput value than all state of the art link fault tolerant strategies. With a redundant section in each link, the NoC saturation throughput can be largely improved than just utilizing FS, e.g. 18% when 10% of the NoC wires are broken. Simulation results we obtained at various wire broken rate configurations indicate that we achieve the highest saturation throughput if 4- or 8-section links with a flit transmission latency longer than 4 cycles are deactivated.**

***Index Terms*—Networks-on-Chip, partially faulty link utilization, redundant section, link deactivation threshold.**

I. Introduction

IT is anticipated that with further transistor dimension scaling as well as packaging innovation, the number of transistors integrated in a chip will keep increasing, following the trend expressed as Moore's Law. In order to efficiently utilize this ever-increasing transistor budget to meet customers' demands for higher performance and lower power consumption, multi-/many-core systems are widely investigated [1]. However, as core-count per chip grows, the low scalability of traditional inter-core communication architectures, e.g., shared buses and crossbars, becomes a critical issue that limits the performance of multi-core systems [2]. In such scenario, a Network-on-Chip (NoC), which can better take advantage of the packet switching technology scalability [3], became the de facto on chip interconnection solution.

C. Chen and S. Cotofana are with the Department of Computer Engineering, Delft University of Technology, 2628 CD, Delft, The Netherlands, e-mail: changlinchen@outlook.com, S.D.Cotofana@tudelft.nl.

Yaowen Fu is with the Department of Electronic Science and Engineering, National University of Defense Technology, 410073, Changsha, Hunan Province, P.R. China. e-mail: fuyaowen@sina.com

Nevertheless, transistor miniaturization not just improves the chip capability, but also makes the manufacturing yield and chip dependability increasingly serious concerns. As the transistor size decreases, the expectation of getting fault-free chips from the manufacturing process reduces significantly due to issues like manufacturing defects [4] and Process Parameter Variations (PPV) [5] [6]. Meanwhile, permanent faults are more likely to happen at runtime as the chip wear-out speed also increases [7]. Thus besides applying novel physical design strategies to improve the chip yield, it is also necessary to detect and tolerate permanent faults at runtime.

We note that even though the permanent faults can happen at any place in the chip in this paper, we only focus on faults in the NoC since it geometrically spreads all over the chip real estate. We assume that the other components of the chip, e.g., the Processing Units (PUs), are already protected by dedicated fault tolerant strategies. In many cases, the router defects can be perceived as link defects, i.e., a malfunctioning router port can be treated as the link incident to it being broken. In view of this, we only focus on permanent link faults in the remainder of the paper.

A NoC link is composed by a number of wires, which are responsible for transmitting data and control bits between two routers. The ratio of broken wires amount over the link width is the metric to characterize the link fault level, which is a measure of its available bandwidth. Given that the number of permanent chip faults, and by implication the amount of broken link wires, is increasing with chip aging, an irreversible augmentation of the link fault level takes place during chip lifetime. As faults may dramatically degrade the system performance and eventually render the chip useless, they should be tolerated to ensure graceful performance degradation as discarding the chip due to some interconnect faults may be economically unaffordable.

In this line of reasoning links with different fault level should be treated with the following strategy: (i) utilize partially faulty links with low fault level rather than treat them as totally broken, and (ii) declare links with high fault level as broken and make use of a Fault Tolerant Routing Algorithm (FTRA) to route packets along alternate paths.

To reach this target, we make the following contributions in this paper:

- We divide the links into a number of equal width sections and utilize simple test vectors to diagnose the links status at section level, i.e., a link section is deemed as broken if it contains faulty wires. Link sections that were disabled by intermittent errors can be reactivated when the intermittent errors vanish.

- We proposed a Flit Serialization (FS) strategy to utilize partially faulty links with low fault level. We introduces fault-tolerant transmitters and receivers, placed inside the output and input ports of NoC routers, respectively, to make use of all fault-free link sections while mitigating the unbalance between the flit size and the actual link bandwidth. Due to this misalignment flit sections are serialized at the transmitter side to fit the narrowed link, and are deserialized at the receiver side to reconstruct integral flits. The proposed transmitter and receiver are transparent to the router such that their utilization is not constrained by router architecture and implementation or network topology.
- We propose the link augmentation with one redundant section as a low cost mechanism to mitigate the FS drawback that a link's available bandwidth is reduced even if it contains only 1 faulty wire.
- We deactivate Heavily Defected (HD) links whose fault level exceed a certain threshold. The optimal threshold is derived by comparing the zero load packet transmission latency on the HD links and that on the shortest alternative path. Deactivated links are dealt with a Fault Tolerant Routing Algorithm (FTRA) which can also efficiently utilize UnPaired Functional (UPF) links in the partially broken interconnects.

The proposed link fault tolerant architecture is evaluated with both synthetic traffic and application tracks from the PARSEC [8] benchmark suite in the context of a baseline NoC system implemented in Verilog HDL at RTL level. We compared our approach against equivalent state of the art solutions, i.e., the spare wire replacement method [9], the Partially Fault Link Recovery Mechanism (PFLRM) [10], and the Simple Flit Half Splitting (SFHS) method [11]. Experimental results indicate that the FS method can achieve lower area*power/saturation_throughput value than all counterpart link fault tolerant strategies. Moreover, with a redundant section in each link, the NoC saturation throughput can be largely improved than just utilizing FS, e.g. 18% when 10% of the NoC wires are broken. Simulation results we obtained at various wire broken rate configurations indicate that we achieve the highest saturation throughput if 4- or 8-section links with a flit transmission latency longer than 4 cycles are deactivated.

The rest of the paper is organized as follows. Section II presents a brief related work survey. Section III introduces the flit serialization and the link section augmentation strategy. Section IV discusses the link deactivation threshold and how to tolerate deactivated links by means of fault tolerant routing. Section V evaluates the performance of the proposed strategy and compares it with tightly related work. Section VI concludes the presentation.

II. Related Work

Targeting different fault scenarios, numerous strategies have been proposed to utilize partially faulty links and route packets in awareness of link bandwidth variations.

A. Partially Faulty Links Utilization

Intuitively, we can prefabricate spare wires in the chips to replace faulty wires. Grecu et al. [4] use this method to enhance the NoC interconnect yield by mapping m interface signals to n link wires ($n \geq m$). However, their approach is only applied to the manufacturing process and cannot address runtime failures [9]. To address this drawback, Lehtonen et al. proposed a set of runtime faulty wire detection and replacing strategies. In [12], they divide the link into a certain number of sections and provides each section with one spare wire. However, this approach cannot tolerate the case in which more than one faulty wire exist in the same section. In [9], an improved method was proposed where the spare wires are shared rather than exclusively owned by each section. The spare wire replacement method can preserve the original link bandwidth, but the control logic is complicated and thus induces high silicon area cost.

A packet rebuilding/restoring algorithm is proposed by Yu et al. [13] to utilize links with reduced bandwidth. Each link is split into a big part with m bits and a small part with n bits ($m > n$). When a link is defected, the fault free wires in the small part are utilized to replace the faulty wires in the big part. Accordingly, a packet first transmits the most significant m bits of each flit. The non-transmitted small parts of the flits are reassembled into one or more m-bit flits and then transmitted. When more than n wires are broken, the link is abandoned. This method can be seen as implementation of the spare wire replacement method without using prefabricated spare wires. However, because an integral flit can only be restored after all the reassembled flits have been received, this method cannot be utilized in low latency routers with wormhole or Virtual-Cut-Through (VCT) switching technology.

By noticing that the faults are rarely clustered when they randomly happen, Vitkovskiy et al. [10] introduced a Partially Faulty Link Recovery Mechanism (PFLRM), which is mainly comprised of a flit shifter, a de-shifter, and a flit re-assembler. Assuming the maximum fault cluster size is k in a link, it requires $k + 1$ cycles to successfully transmit a flit. In the first cycle, the flit is transmitted in the normal way. In each of the rest cycles, the flit is rotated by 1-bit before being transmitted, thus the data bits that were transmitted on faulty wires in the previous cycle can be transmitted on fault free ones. At the receiver side, the flits are de-rotated and the newly transmitted data bits are selected to reassemble the original flit. Although this approach can theoretically work for defective links with an arbitrary large number of faulty wires, the induced transmission latency overhead can be significantly high. We note that even only a single faulty wire exists in the link, two cycles are needed to transmit a flit successfully.

The aforementioned strategies rely on the exact knowledge of each link wire status, which may induce high burden to the Built-In-Self-Test (BIST) mechanism. Conversely, Palesi et al. [11] and Lehtonen et al. [9] proposed the method of using flit splitting to tolerate faulty wires. In this approach, a link is divided into four sections. The fault-free sections are utilized to transmit flits and the ones containing broken wires are abandoned. Thus the link status is diagnosed at section level,

which reduces the BIST delay overhead and complexity. Note that the link can be divided into more sections to tolerate more broken wires. However, their approaches cannot utilize all fault free link sections. For example, when one of the four link sections is broken, only two functional sections are utilized to transmit flits half by half. In the remainder of this paper, we name this method as Simple Flit Half Splitting (SFHS).

In summary, spare wires can preserve the NoC performance but introduce a high silicon overhead, while SFHS and PFLRM have low area overhead but induce high extra latency. By comparison, our FS method significantly reduces the latency, when compared with SFHS and PFLRM, while maintaining a more reasonable silicon cost, when compared with the spare wire replacement methods.

B. Link Bandwidth Aware Routing

Utilizing defective links with low fault level can preserve NoC bandwidth and avoid severe performance degradation. Moreover, if the underlying routing algorithm is adaptive, a proper path selection strategy can be applied to determine the path with the lowest data transmission delay. However, Heavily Defected (HD) links may cause severe congestion in the upstream routers, and thus should be discarded.

In [11], Palesi et al. proposed an application specific routing function with a set of selection policies which are aware of the link fault distribution. At each routing hop, the best admissible output port is selected with a probability determined according to the link fault level and the traffic conditions. The probability for each link is off-line computed and stored in a routing table. This strategy provides the best system performance when executing a certain specific application. However, it requires complicated off-line computations and accurate traffic analysis, which makes it not suitable for dynamically changing systems.

In [10], Vitkovskiy et al. proposed a path selection strategy which always chooses the next progressive hop that has maximum available Virtual Channel (VC) amount and minimum Effective Link Utilization (ELU). The ELU of a link is the product of the number of flits that traverse this link and the flit transmission latency on the link. For example, a fault free link and a link with a flit transmission latency of 2 cycles have the same ELU if 50 and 25 flits are transmitted via each link, respectively, in the same time period. Vitkovskiy et al. also discussed the impact of discarding HD links on the system performance. However, they did not propose a method to decide if a defected link should be discarded or not.

When defective links are utilized, they exhibit longer data transmission latency than fault free links. Thus all the delay or bandwidth aware Routing Algorithms (RAs), e.g., [10], [14]–[17], can be employed to select the optimal routing path. Such algorithms usually select the best output port from multiple admissible ones according to factors like the output link bandwidth, the number of free VCs in the downstream routers, and the path latency to the destination achieved by means of the Q-learning method [14].

Nevertheless, for some minimal path adaptive RAs, e.g., Opt-Y [18], when the packets are already in the same column or row with the destination routers, there is only one admissible output path and which is utilized regardless of the fault level of the path links, even if discarding the HD links and detouring the packets could be a better option. Thus it is necessary to determine in which conditions HD links should be deactivated.

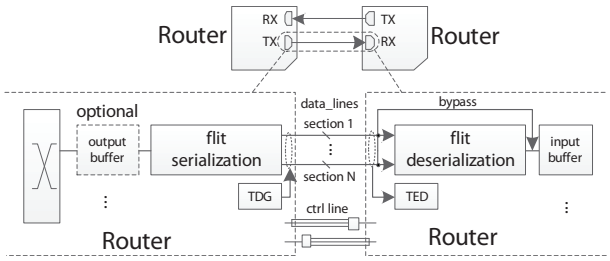


Fig. 1. Proposed fault-tolerant link architecture.

III. Partially Faulty Link Utilization

The principle of the Flit Serialization (FS) strategy is to divide the links and flits into k equal width sections, and make use of all functional link sections to do data transmission. We note that k is preferred to be a power of 2 for the sake of control logic simplicity. Fig. 1 depicts the proposed fault tolerant link architecture. For each unidirectional link, we use a Test Data Generator (TDG) at the Transmitter (TX) side and a Test Error Detector (TED) at the Receiver (RX) side to diagnose the link status and generate a k-bit fault vector to indicate the faulty link sections. If faulty wires exist in at least one link section, sections of adjacent flits are serialized by the flit serialization unit at the TX side and then transmitted on the fault free link sections. All flit sections are then deserialized at the RX side to reconstruct the original flits. On fault free links, the flits are transmitted according to the normal protocol, bypassing the proposed flit serialization and deserialization units. The FS mechanism is transparent to the rest of the router parts thus its utilization is not constrained by the router architecture and implementation, or by the network topology.

As the number of control signals, e.g., the data valid signal and the credit control signals, in each link is much smaller than the number of data lines, they are protected by Triple Modular Redundancy (TMR) method with a marginal silicon area overhead. If a Error Correcting Code (ECC) is utlized to protect data from transient errors, the error coding logic should be placed before the flit serialization unit and the error decoding logic should be placed after the flit deserialization unit, thus soft errors generated in the data link as well as inside the transmitter and/or the receiver can be detected and corrected.

A. Link Diagnosis

Unlike spare wire replacement and PFLRM, which need to know the precise status of each wire, our method just needs link fault vectors at the section level, i.e., a link section is broken if it contains broken wires. For example, if the third section of a link (with 4 sections in total) contains faulty wires, the fault vector of the link is marked as “1101”. Thus for an n-bit wide link divided into k sections, we just need a k-bit wide register to store the section level fault vector.

In this paper, we assume that the possible permanent faults are stuck-at, i.e., the wire value is stuck at '1' or '0', and crosstalk, i.e., the status of two adjacent wires interfere with each other and one is in the dominant position and determines the value of the other one. Under these fault models, to detect the faults in a 4-bit wide link section, the TDG sequentially injects 2 test vectors "0101" and "1010" to the link section under testing. At the RX side, received data are XORed with expected values and if the result is not always "0000", an error signal is asserted to indicate that faults exist in the link section. As a comparison, to achieve bit level fault vector, the link diagnosis method in [10] uses the same test vectors but can only detect stuck at faults, the method in [19] can detect crosstalk faults but requires 8 test vectors and thus 8 clock cycles to diagnose 1 wire, and the method in [9] can diagnose wires status without ceasing data transmission but requires complicated control logic. We note that to distinguish permanent errors from soft ones, the test is repeated 3 times and the link section is marked as broken only when the error signal is asserted at least twice.

Link diagnosis is triggered i) periodically and ii) when the number of soft errors detected by the ECC logic exceeds a predefined threshold in a short time period [9]. To avoid draining the NoC completely, the links are diagnosed one after another in case i) and only the links with high soft error frequency are diagnosed in case ii). In collaboration with the fault tolerant routing algorithm, the links under diagnosis can be temporally treated as broken and the packets are detoured along alternative paths. Because intermittent errors, e.g., crosstalk faults that only happen at certain temperature/voltage, may have the same syndrome as permanent errors when they happen, sections which are marked as faulty in the previous test are also tested, to prevent situations when vanished intermittent errors are still disabling sections. At the end of the diagnosis process the achieved fault vector is sent to the transmitter via a TMR protected serial wire.

B. Flit Serialization and Deserialization

In a typical NoC router, the head flit of each packet has to sequentially go through 3 pipeline stages: Routing Computation (RC), combined Virtual Channel (VC) Allocation (VA) and Switch Allocation (SA), and Crossbar Transversal (CT). An extra Link Transversal (LT) stage is usually required to send flits to the downstream router. The proposed flit serialization and deserialization units are implemented in the LT stage. For the sake of simplicity, we present our proposal for the case when both flits and links are divided into 4 sections, and the link width is 40-bit, i.e., each link section is 10-bit wide. We note that the proposed principle is more general and can be applied to wider links with more sections.

Fig. 2 illustrates the structure of the proposed flit serialization unit. In general, each output port embeds a 1-flit width link register to store flits before they are sent to the downstream router. To allow for flit serialization, we expand the link register width to 2-flits and divide it into 8 sections that can be read and write independently. The register is designed in such a way that a new flit can be registered in the Least Significant Half (LSH) of the register if there are

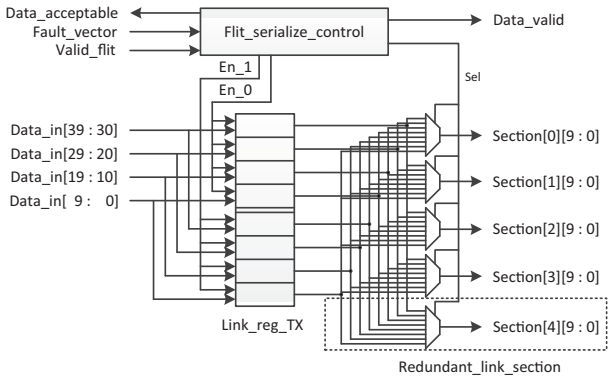


Fig. 2. Flit serialization unit - TX.

flit sections in its Most Significant Half (MSH) still waiting to be transmitted, and vice versa. If the link is fault-free, only the register MSH is utilized, acting as a conventional link register. Otherwise, flits are serialized under the control of the *flit_serialize_ctrl* unit. The serialization process is presented in more detail in Section III-C. The number of flit sections that can be transmitted in each cycle is the same as that of fault free link sections. The *Data_valid* signal indicates the downstream receiver whether valid data are transmitted on the link in each cycle.

Multiplexers are utilized to select the to be transmitted flit sections. Each selector's value is determined by an adder according to (1).

$$sel[i]\,(t) = (sel[i]\,(t-1) + k_{fault\_free})\,\%2k \qquad (1)$$

where k_{fault_free} is the number of fault free sections. If the third section of a 4-section link is broken, the initial values of sel[0]-[3] are 0, 1, X, and 2, respectively. If k is a power of 2, the mod operation can be removed by properly choosing the adder width.

With a narrowed link, the flit is transmitted on the link at a lower rate than on the router crossbar and we rely on the *data_acceptable* signal to indicate if the next flit can be accepted by the serialization unit subject to the remained buffer space. The *data_acceptable* generation logic is also easier to implement if k is a power of 2 than other value as it needs to refer to the multiplexer selectors' value.

At the RX side, a flit deserialization unit (see Fig. 3) reconstructs the flit out of the serialized sections. Similar with the flit serialization unit, a 2-flit wide link register (*link_reg_RX*) is employed. Multiplexers are utilized to select the valid sections from the link under the control of the *flit_deserialize_ctrl* unit. The newly received flit sections are stored into the correct link register position to reassemble integral flits. When the link register MSH or LSH is full, one flit was assembled and can be read out by the router.

C. Flit Transmission Process

Fig. 4 graphically depicts the timing diagrams capturing the flit transmission process specific to our method. When the link is fault-free, a flit from the crossbar is loaded into the *link_reg_TX* MSH and then directly transmitted to the downstream router input buffers (see Fig. 4(a)). At the RX side, the flit deserialization unit is bypassed.

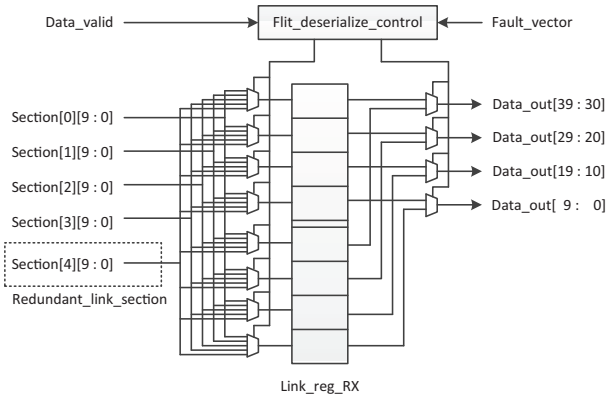


Fig. 3. Flit deserialization unit - RX.

We introduce the FS method with an example situation when one of the 4 link sections is affected by faults. As illustrated in Fig. 4(b), at TX side, flit a floats at the output port of the crossbar at the rising edge of $T1$ and is written into the *link_reg_TX* MSH at the rising edge of $T2$. During the $T2$ cycle, the first three sections of the flit a (a_3, a_2, a_1) are transmitted to the downstream router via the three fault-free sections of the link. At the rising edge of $T3$, flit b is written into the LSH of *link_reg_TX*. Flit sections a_0, b_3, and b_2 are transmitted in the same cycle. The signal *data_acceptable* is set to '0' in $T3$ such that no new flit may appear at the crossbar output port in $T4$. A wait cycle is inserted to allow for the transmission of the last three sections of flit c during $T5$. The signals *high_reg_state* and *low_reg_state* are utilized to indicate the status of *link_reg_TX* MSH and LSH, respectively. Each signal is asserted once a flit is written into the corresponding register part, and de-asserted in the clock cycle when all data belonging to the flit are read out.

At the receiver side (see Fig. 4(c)), flit sections are de-serialized and reassembled into integral flits in *link_reg_RX*. Valid flit sections are selected by input side demultiplexers and written at the correct positions in *link_reg_RX*. Once the register MSH or LSH is full, an integral flit is recovered. The signals *flit_1_recovered* and *flit_2_recovered* indicate the availability of recovered flits and control the output side multiplexers to select the corresponding register sections.

D. Redundant Link Section

Even if we can efficiently utilize the remained link bandwidth, the flit transmission latency is increased when faults exist. As illustrated in Fig 2 and Fig. 3, by extending each link with one redundant section, we can combine the benefits of the spare wire replacement method and the FS method. If only one fault exists, or multiple faults exist but "luckily" they are all resident in the same link section, the link can still transmit one integral flit per cycle. We note that such scenario is a "disaster" for the PFLRM method as it may cause large fault cluster size and hence long flit transmission latency.

We do not rely on bit level spare wire replacement methods, e.g., [9], [12], because overlapping FS with these methods requires an extra multiplexing and demultiplexing step, and by implication of more multiplexers, demultiplexers, and selection bit registers, which can significantly increase the link critical path length, area cost, and power consumption.

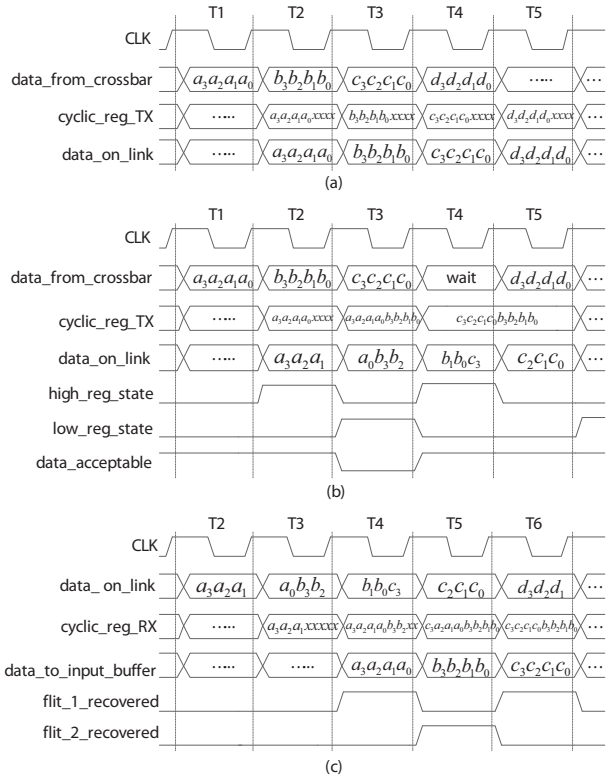


Fig. 4. Timing diagram of proposed mechanism (a) Timing diagram for a fault-free link; (b) Transmitter side when one section contains faulty wires; (c) Receiver side when one section contains faulty wires.

Given that FS extra area is dominated by the 2-flit wide link registers, especially for wide links, and that adding one redundant link section does not require larger link registers the link augmentation with a redundant section is a cost effective solution. As illustrated in Section V, by adding one redundant section per link we can achieve up to 18% saturation throughput improvement than just utilizing FS when the link fault rate is as high as 0.1, with only 4.7% area and 2.4% power overhead, respectively.

E. Flit Transmission Latency and Reliability

The link flit transmission latency when the FS method is utilized (l_{FS}) to continuously transmit a number of flits (*flit_number*) can be expressed as:

$$l_{FS} = \left\lceil \frac{k \times flit\_number}{k_{fault\_free}} \right\rceil, \tag{2}$$

For example, transmitting 10 flits via a link which has one broken section requires 14 and 12 cycles when the link is divided into 4 and 8 sections, respectively.

For the sake of comparison, PFLRM (l_{PFLRM}) and SFHS (l_{SFHS}) flit transmission latencies are expressed in (3) and (4), respectively.

$$l_{PFLRM} = (cluster\_size + 1) \times flit\_number, \tag{3}$$

where *cluster_size* is the maximum fault cluster size.

$$l_{SFHS} = \frac{k}{k_{available}} \times flit\_number. \tag{4}$$

Note that the $k_{available}$ can be equal with or less than the number of fault free sections in the link and can have the value

TABLE I
AVERAGE FLIT TRANSMISSION LATENCY (CYCLES/FLIT) WHEN FLITS ARE TRANSMITTED CONTINUOUSLY

number of faults		0	1	2	3	4	5	6	7	8
FS	S4	1	1.33	2	4	–	–	–	–	–
	S8	1	1.14	1.33	1.60	2	2.67	4	8	–
PFLRM		1	2	3	4	5	6	7	8	9
SFHS	S4	1	2	2	4	–	–	–	–	–
	S8	1	2	2	2	2	4	4	8	–

of $2^i, i = 0, 1, 2, ...$ For example, when the link is divided into 4 sections, it can be 1, 2, or 4.

Table I presents the average flit transmission latency (cycles/flit) when FS, PFLRM, and SFHS are utilized to continuously transmit flits via a defective link. The number of faults in the first table row indicates the fault cluster sizes for PFLRM, and the numbers of faulty sections for FS and SFHS. S4 and S8 mean that the link is divided into 4 and 8 sections, respectively. From the Table we observe that PFLRM and SFHS latencies double in the presence of one error while for FS this happens only after half of the link sections are broken.

In Fig. 5, we depict the average flit transmission latency on 40-bit wide links when fault wires are uniformly distributed and each link is divided into 8 sections. The results are obtained by doing Monte Carlo simulations when the following methods are applied: FS, FS with one redundant link section (FS+1), PFLRM, and SFHS. Note that the links with no fault free section are not considered. We can observe that from the statistic point of view, FS provides lower flit transmission latency than PFLRM when the link has less than 6 faulty wires. When more faulty wires exist, FS performs worse than PFLRM but still better than SFHS. We can also observe that FS+1 achieves lowest flit transmission latency when there are less than 9 faulty wires.

When 9 or more faulty wires are uniformly distributed in the links, PFLRM outperforms all the other counterparts. However, in the cases corresponding to large physical defects multiple, e.g., k, adjacent wires may get faulty. In such a scenario, PFLRM requires $k+1$ cycles to successfully transmit a flit, which results in a large latency overhead, and a spare wire replacement method has to make use of k spare wires, which results in a large area overhead. Given that such a large defect will most likely affect only one or two link sections the proposed FS approach can handle such extreme cases with an efficiency corresponding to the case when one or two faulty wires are detected in the link.

In principle, we can split a link into more sections, e.g., 16 or more, to achieve more graceful performance degradation. However, this implies that more and larger multiplexers are required, which have a negative impact on area and power overheads. If we assume that each wire has the same probability (p_e) to be permanently faulty, the probability that an n-bit wide link has k faulty wires can be calculated using (5).

$$P_k = \binom{n}{k} {p_e}^k (1 - p_e)^{n-k} \quad (5)$$

Thus even if p_e is as high as 0.001, the probabilities for a 50-bit wide link to have 4 and 8 faulty wires are only

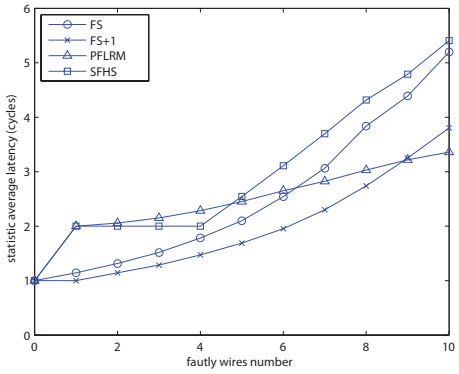


Fig. 5. Average flit transmission latency of different partially faulty link utilization strategies. The link is divided into 8 sections for FS and SHFS.

2.2×10^{-7} and 5.1×10^{-16}, respectively. We note that the average flit width of most state of the art NoCs is 50.1-bits [20]. This means that by dividing the link into 8 sections, we can ensure that the link probability to have a flit transmission latency of 2 cycles is lower than 10^{-7} and the probability for a link to be totally broken is lower than 10^{-16}. Given that faulty wires are not always evenly distributed in different sections, the aforementioned two probabilities are much lower in practice. In view of this analysis, we conclude that dividing links into more than 8 sections has no practical relevance for most state of the art NoCs, and the section number should be a power of 2, e.g., 4 or 8, to achieve simple control logic. The actual number of sections can be determined via a trade-off process which takes into consideration the targeted fault-tolerance capability and the available silicon real estate.

IV. HEAVILY DEFECTED LINKS TOLERATION

Utilizing defective links that have low fault level can partially preserve the NoC link bandwidth and reduce the transmission latency overhead caused by packet detouring and congestion, while utilizing Heavily Defected (HD) links may cause server congestion in the upstream routers. Whether to make use or not of a defective link can be decided (i) dynamically based on the local or global traffic load or (ii) statically by checking if its fault level has exceeded the link deactivation threshold. In case (i), the Routing Algorithm (RA) selects the best output port according to factors like output link bandwidth, the number of free VCs in the downstream routers, and the paths latency to the destination which is achieved by means of the Q-learning method [14]. Conversely, the static solution, i.e., case (ii), just requires the calculation of the appropriate link deactivation threshold value, and lets each router to decide whether a link incident to it should be deactivated. The calculation can be performed off-line according to the link structure, traffic pattern, and the underlying RA. Due to the fact that FS induced flit transmission latency increases slowly when the link fault level is low and fast when the fault level is high, it is easy to determine the optimal threshold and thus we choose to rely on the static solution in our proposal. We note that when the static solution is utilized, the link bandwidth and delay aware path selection strategies, e.g., [10], [11], [15], can still be applied to select the best output port.

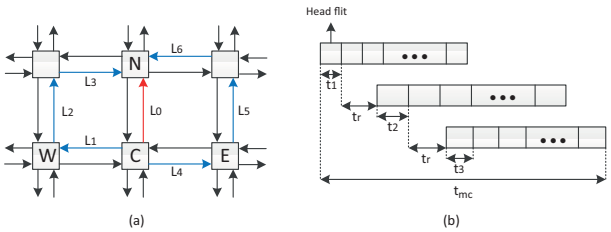


Fig. 6. Detouring example. (a) The misrouting-contour of L_0. (b) Detouring delay.

A. Link Deactivation Threshold

From (2) we can observe that for the FS strategy, the link flit transmission latency is inversely proportional with the number of fault free link sections. Thus for a k-section link, when the number of broken sections increases from b to $b+1$, the flit transmission latency on this link becomes $(k-b)/(k-b-1)$ times higher. For example, when each link is divided into 8 sections, the flit transmission latency is only increased by 14% when a new section of a fault free link becomes broken, while the latency is doubled when the number of broken link sections increases from 6 to 7. This FS property makes it easy to decide the optimal link deactivation threshold.

Take the case illustrated in Fig. 6 for example. Assuming a packet is waiting to be transmitted from router C to router N. By default, the packet should be transmitted via L_0. The question now is: If L_0 is defected, at which fault level should we abandon it and detour the packets to achieve minimum system performance degradation?

If L_0 is deactivated, most probably the packets will be misrouted along alternative paths formed by its adjacent links, i.e., $L_1 \rightarrow L_2 \rightarrow L_3$, or $L_4 \rightarrow L_5 \rightarrow L_6$. We refer to such paths with the concept of *misouring-contour*. According to the outgoing direction of L_0, we can divide its misrouting-contour into the left half, i.e., $L_1 \rightarrow L_2 \rightarrow L_3$, and the right half, i.e., $L_4 \rightarrow L_5 \rightarrow L_6$.

Let us assume that the packet length is P, the NoC operates according to the wormhole switching technique [21], and each router has 3 pipeline stages. At zero traffic load, the time required to transmit an entire packet to router N via L_0 (T_{L_0}) and its left half misrouting-contour (T_{mc}) can be expressed as (6) and (7), respectively. Here we assume that the packets are detoured along the left side misrouting-contour by default, but we note that the analysis to the right half misrouting-contour can be done in a similar way.

$$T_{L_0} = Pt_0, \tag{6}$$

$$T_{mc} \geq t_1 + t_r + t_2 + t_r + Pt_3, \tag{7}$$

where t_i (≥ 1 cycle) is the flit transmission latency on link L_i, i=0,1,2,3, and t_r (≥ 3 cycles) is the latency for a head flit to traverse a 3-stages pipelined router. In (7), T_{mc} is larger than the right side polynomial when t_1 or t_2 is large enough to create the situation that a flit arrives at a router input port after all precedent flits have already been transmitted to the next hop.

Obviously, we should deactivate L_0 and detour the packets on the misrouting-contour when

$$T_{L_0} > T_{mc} \geq 8 + P, \tag{8}$$

i.e.,

$$t_0 > (t_1 + t_2 + 6)/P + t_3 \geq 8/P + 1. \tag{9}$$

Thus the minimum link deviation threshold is inversely proportional to the packet length, e.g., the threshold is $t_0 > 3$ cycles when the packet length is 4 flits and decreases to $t_0 > 1.5$ cycles when the packet length is 16 flits. However, in practice, T_{mc} is much higher than $8 + P$ due to the fact that: (i) the links on the misrouting-contour are not always fault free, (ii) detouring the packets increases the congestion on the misrouting-contour especially at high traffic load, and (iii) extra flow control delay [21] should be considered in T_{mc}.

We note that at near zero traffic load, the possibility for a detoured packet and a normally transmitted packet to compete for the same NoC resource is low, case in which deactivating HD links at the minimum threshold can reduce the packet transmission latency. However, the congestion on the misrouting-contour increases as the traffic load gets higher, thus if L_0 is deactivated at the minimum threshold, the packet transmission latency on the misrouting-contour can easily surpass the one on L_0 at moderate traffic load. This implies that in practice the link deactivation threshold should be set higher than the minimum one. In fact, even at low traffic load, moderate increase of the link deactivation threshold will only bring negligible increase of the average packet transmission latency because the FS induced flit transmission latency increases slowly when less than 75% of the sections cannot be utilized. Thus L_0 should be deactivated only when $T_{L_0} \gg T_{mc}$. In view of such analysis, we adjust the link deactivation threshold to $t_i > 4$ cycles for both short and long packets, i.e., an 8-section link is deactivated when it has 7 or more broken sections, and a 4-section link is deactivated when all link sections are broken. The effectiveness of the selected threshold is validated in Section V-C.

B. Fault Tolerant Routing

Conventionally, the interconnection between two adjacent NoC routers comprises a pair of unidirectional links, each link having its own control flow wires and handling either outgoing or incoming traffic. If one unidirectional link is broken or deactivated, one data transmission direction is lost and the packets have to be detoured. Tsai et al. [22] suggest to replace the unidirectional links with bidirectional ones such that when one link is broken, the other one can be utilized in both directions resulting in a half-duplex communication. However, unidirectional links are still attractive as they provide better means to implement the control logic and to address timing error issues [23]. Besides, when an input or output port is broken, a bidirectional link also becomes unidirectional. In view of these observations, in this paper, we focus on NoCs that utilize unidirectional links.

As the two unidirectional links in one interconnection are physically independent from each other, it is highly possible that when one link is deactivated, the other one is still utilized and becomes UnPaired Functional (UPF) link. To efficiently utilize UPF links in NoC, we rely on the UPF link aware FTRA (UPF-FTRA in short) we proposed in [24]. UPF-FTRA is able to utilize all UPF links that are incident to active

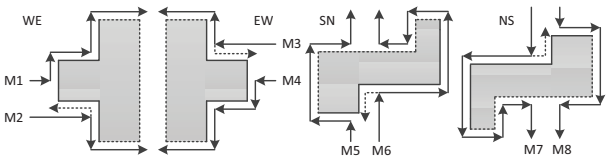


Fig. 7. Misrouting direction of different messages. The dashed boarder of the shadows may not be fault walls.

routers. The basic fault pattern tolerated by UPF-FTRA is a fault wall, which is composed of adjacent broken links with the same outgoing direction. More complicated fault regions can be formed by multiple fault walls with different fault directions. Messages are routed around the fault walls along the misrouting contours of the broken links. UPF-FTRA requires at least 3 Virtual Channels (VCs) and dynamically reserves them to the messages whose default transmission path is blocked to guaranty deadlock freeness. Please refer to [24] for the proof that UPF-FTRA is deadlock free. We note that although various FTRAs have been proposed, most of them, e.g., [10], [25]–[33], abandon the entire interconnection without trying to take advantage of the available UPF links. Thus the UPF links are wasted even though utilizing them can partially preserve the link capabilities and can result in graceful system performance degradation.

With UPF-FTRA, the packets misrouting path is known once the NoC fault pattern is validated. According to the outgoing direction of a broken link, the default packet misrouting direction is clockwise, i.e., along the left side misrouting-contour. The counter-clockwise misrouting direction is utilized only when with the default misrouting direction row messages will be forced to return to the previous column or column messages will be forced to return to the previous row by another fault wall or NoC edge. For example, in Fig. 7, messages M1, M4, M5, and M8 are misrouted in clockwise direction, while messages M2, M3, M6, and M7 are misrouted in counter-clockwise direction as otherwise they will be forced to return to the previous column or row as suggested by the dashed arrows.

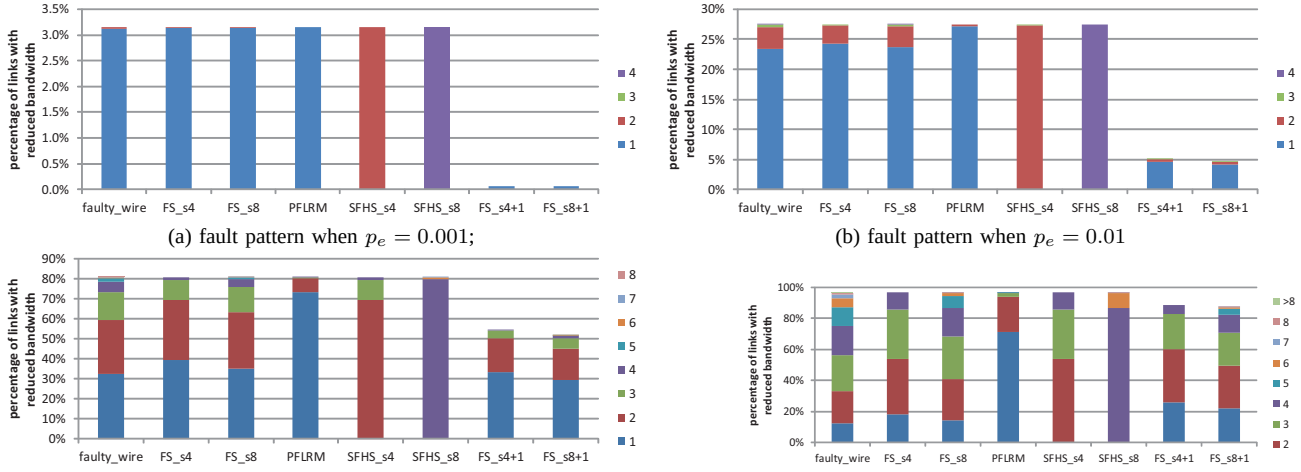
V. Evaluation

To put the implications of our link fault-tolerant architecture in a better practical prospective, we evaluate and compare it with other three tightly related proposals presented in [9], [10], and [11], namely spare wire replacement, PFLRM, and SFHS, respectively. To this end, we implemented all these four link fault-tolerant methods at RTL level by using Verilog HDL, and applied them in the context of an 8×8 2D mesh NoC platform developed by Lu et al. [34]. The NoC employs wormhole switching technique and credit based flow control mechanism. Each baseline router has 3 pipeline stages, i.e., RC, VA/SA, and CT, and 5 Physical Channels (PC). Each PC is shared by 4 VCs, and each VC buffer is 4-flit deep and 40-bit wide, as both of flit and link widths are 40-bit. As UPF-FTRA requires 3 VCs to be deadlock free, the 4th VC is freely utilized by any message type without causing deadlock. The router and the link fault-tolerant modules are synthesized using the Synopsys Design Compiler with TSMC 65-nm standard cell as target technology. We first evaluate the FS performance with both synthetic traffic and recorded traffic traces from PARSEC benchmarks [8], and then verify the effectiveness of the link deactivation threshold derived in Section IV-B in the context of different fault patterns.

A. FS Performance with Synthetic Traffic

To evaluate the performance of the FS method, we first run synthetic uniform random traffic in the context of different fault link patterns for a wide range of permanent wire fault rates, i.e., 0.001, 0.01, 0.05, and 0.1. We assume that each wire has the same fault rate p_e and faults are uniformly distributed across the links. We note that this is the worst scenario to the FS scheme. Any other fault distribution pattern will cause the faults more clustered, case in which the FS scheme has even better performance as we explained in Section III-E. The three partially faulty link utilization strategies, i.e., FS, PFLRM, and SFHS, are applied to the NoC system and simulated for each fault pattern. Note that when spare wire replacement method is employed, the original NoC performance is preserved until all spare wires are utilized to replace the broken wires. For the section based strategies, i.e., FS and SFHS, we simulated two cases when each link is divided into 4 (FS_s4 and SFHS_s4) and 8 (FS_s8 and SFHS_s8) link sections, respectively. For the FS method we also simulated the case when each link is augmented with one redundant link section, i.e., FS_s4+1 and FS_s8+1. Each packet consists of 4 flits and is routed with the XY routing protocol.

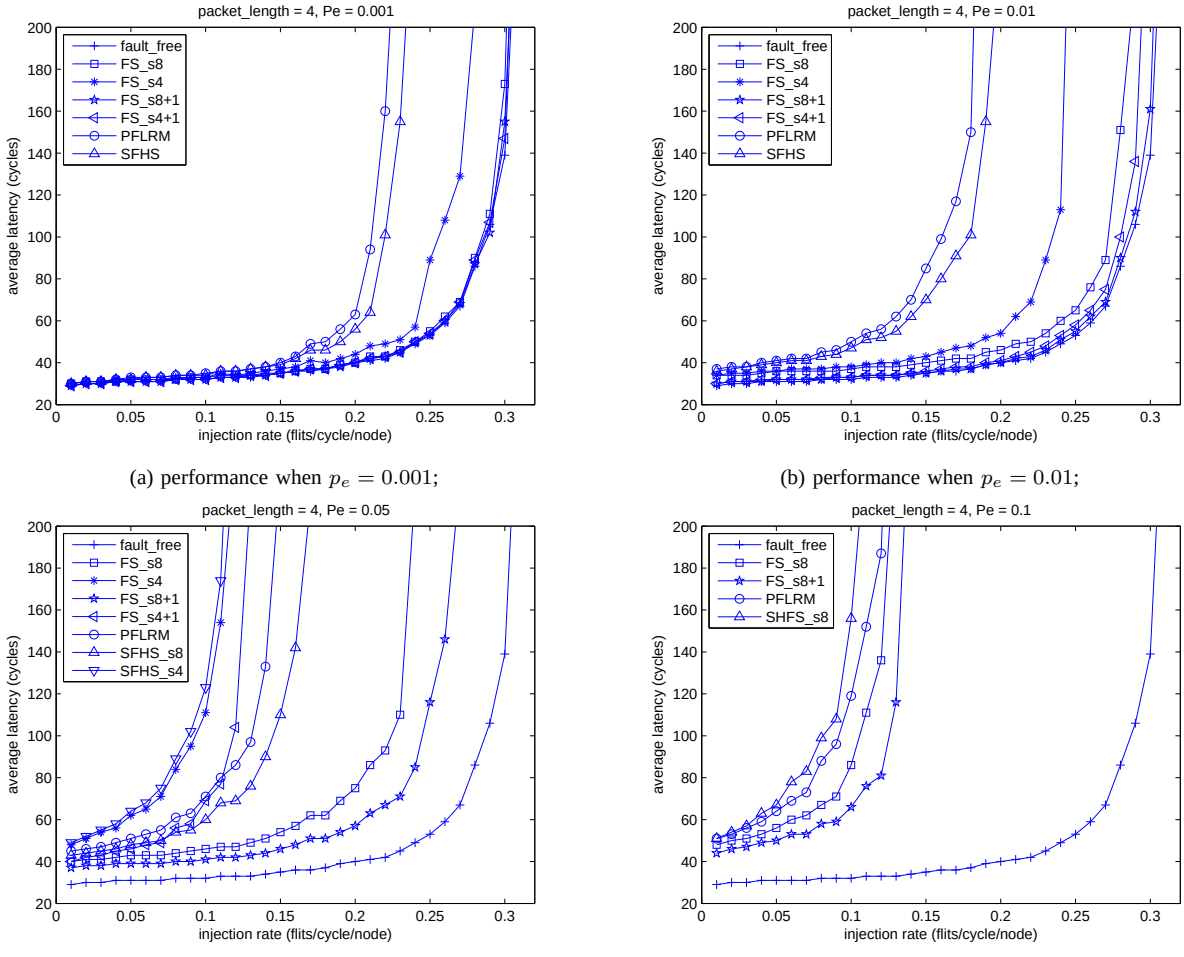
We first run Mento Carlo Simulations to study the fault distribution at different wire fault rates. We randomly create faulty wires in the NoC and count the number of defected links. For each defected link, we count the number of faulty wires, the maximum fault cluster size, and the number of broken link sections. The fault distribution is illustrated in Fig. 8. In the figure, each column's height represents the percentage of defective links present in the NoC. In each column, different colors represent the percentage of defective links with different fault levels. For example, the columns' red parts denote the percentage of links with 2 faulty wires, or links with 2 unusable link sections in FS and SFHS, or links with a fault cluster size of 2 in PFLRM. Take Fig. 8(b) for example, it illustrates that when p_e is 0.01, 27.4% of the NoC links are defected, among which the percentage of links contain 1, 2, and 3 faulty wires are 23.4%, 3.7%, and 3%, respectively. It means that: (i) if each link is divided into 4 sections, the percentage of links contain 1, 2, and 3 broken sections are 24.2%, 3.0%, and 0.2%, respectively; (ii) if each link is divided into 8 sections, the percentage of links contain 1, 2, and 3 broken sections are 23.7%, 3.4%, and 0.3%, respectively; and (iii) the percentage of links have a fault cluster size of 1 and 2 are 27.1% and 0.3%, respectively. For FS with one redundant section per link, the link bandwidth is reduced only when 2 or more sections are broken, thus for FS_s4+1 and FS_s8+1 the percentage of links with reduced bandwidth is much lower than that in the other cases. Note that SFHS has much lower link bandwidth utilization efficiency than FS, e.g., even if a link has only 1 broken section, SFHS_s4 and SFHS_s8 treat it in the same way as 2 and 4 sections are broken, respectively. Such SHFS property is reflected in Fig. 8 by rounding up the actual number of broken link sections to its equivalent cases



(a) fault pattern when $p_e = 0.001$; (b) fault pattern when $p_e = 0.01$

(c) fault pattern when $p_e = 0.05$ (d) fault pattern when $p_e = 0.1$

Fig. 8. Fault link Patterns at different wire fault rate. In the FS_s4+1 and FS_s8+1 cases, one redundant link section is provided, and links with only 1 broken link section are not counted in as they still can transmit one integral flit per cycle.



(a) performance when $p_e = 0.001$; (b) performance when $p_e = 0.01$;

(c) performance when $p_e = 0.05$; (d) performance when $p_e = 0.1$.

Fig. 9. NoC Performance at different wire fault rate.

with more broken sections.

Fig. 9 depicts the NoC performance measured in terms of average packet transmission latency obtained when different partially faulty link utilization strategies are applied. The packet transmission latency is counted since the packet is generated in the source node till the tail flit is received by the destination node, i.e., the queuing time in the source node is included. We gradually increase the Flit Injection Rate (FIR) at a step length of 0.01 flits/cycle/node to derive the near zero traffic load packet transmission latency and the saturation throughput, i.e., the FIR when the packet latency approaches infinity.

1) Without Redundant Link Section: We first compare the performance of the three partially faulty link utilization strate-

gies when the redundant link section is not provided.

As indicated in Table I, FS_s8 induces the lowest flit transmission latency overhead on defective links with 1 broken section, and thus it achieves the best performance when p_e is 0.001, i.e., the average packet transmission latency is very close to the fault-free case (see Fig. 9(a)). At such fault rate, the performance of FS_s4 is lower than that of FS_s8 but still much better than that of PFLRM and SFHS. This can be explained by the fact that in links with one broken section, both PFLRM and SFHS will double the flit transmission latency at least, while FS_s4 can keep the latency overheads as low as 33.3%. Note that for both SFHS_s8 and SFHS_s4, the flit transmission latency on the defective links is doubled at this fault rate thus they have the same performance.

As p_e increases, more links contain faults and the average number of faulty wires becomes larger, leading to more unusable sections and bigger fault cluster size in links. The average flit transmission latency increases for all partially faulty link utilization strategies. But when the fault rate is not very high, e.g., $p_e = 0.01$, FS still outperforms PFLRM and SFHS (see Fig. 9(b)).

When p_e further increases to 0.05, FS_s8 still achieves the best performance because the flit transmission latency on more than 99% of the defective links is less than 2 cycles. However, FS_s4 performs worse than SFHS_s8 and PFLRM. This is because the number of links that have a flit transmission latency higher than 2 cycles in FS_s4 is much larger than that in SFHF_s8 and PFLRM. These slow links cause severe congestion in their upstream routers and hence obvious system performance degradation. We note that at such p_e value, totally broken links can exist in FS_s4 and SFHS_s4. To avoid the implication of FTRAs to the performance of partially faulty link utilization methods, the fault patterns which contain totally broken links are not considered in this subsection. This cannot fundamentally affect the results because only 1 or 2 such links may exist in the NoC at this fault rate.

When the permanent wire fault rate is as high as 0.1, 96.6% of the links are defective and the average fault level is high, as depicted in Fig. 8(d). Under this extreme conditions, FS_s8 exhibits only slightly better performance than PFLRM (see Fig. 9(d)) while FS_s4 and SFHS_s4 have so many totally broken links that they are not considered. If the fault rate keeps on increasing, the average packet transmission latency in FS_s8 increase and eventually its performance gets worse than that of PFLRM.

2) With One Redundant Link Section: As illustrated in Fig. 8, when each link is augmented with one redundant link section, the numbers of defective links when p_e is 0.001, 0.01, 0.05, and 0.1 are reduced by 98%, 82%, 32%, and 8%, respectively. The system performance, in terms of average packet transmission latency and saturation throughput, is also obviously improved. For example, when p_e is up to 0.01, FS_s4+1 and FS_s8+1 can still provide similar performance with the fault free case, while FS_s4 and FS_s8 induce 21% and 3% saturation throughput degradation already. When $p_e = 0.05$, one redundant link section can improve the FS_s4 and FS_s8 saturation throughput by 20% and 8.7%, respectively. Although the number of defective links is only reduced by 8% when $p_e = 0.1$, the FS_s8 saturation throughput is improved by 18%. As the wire fault rate increase from 0 to 0.1, FS_s8+1 provides the most gracefully system performance degradation when compared with other counterpart partially faulty link utilization strategies.

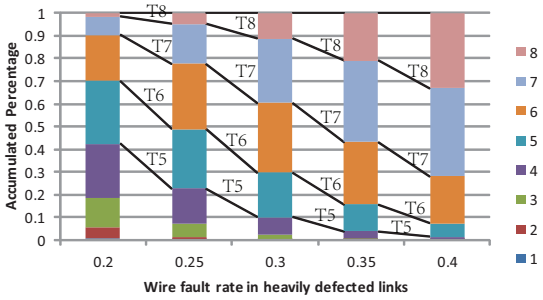


Fig. 11. The link fault level change trend at different wire fault rate. The legend is the number of broken link sections in a heavily defected link.

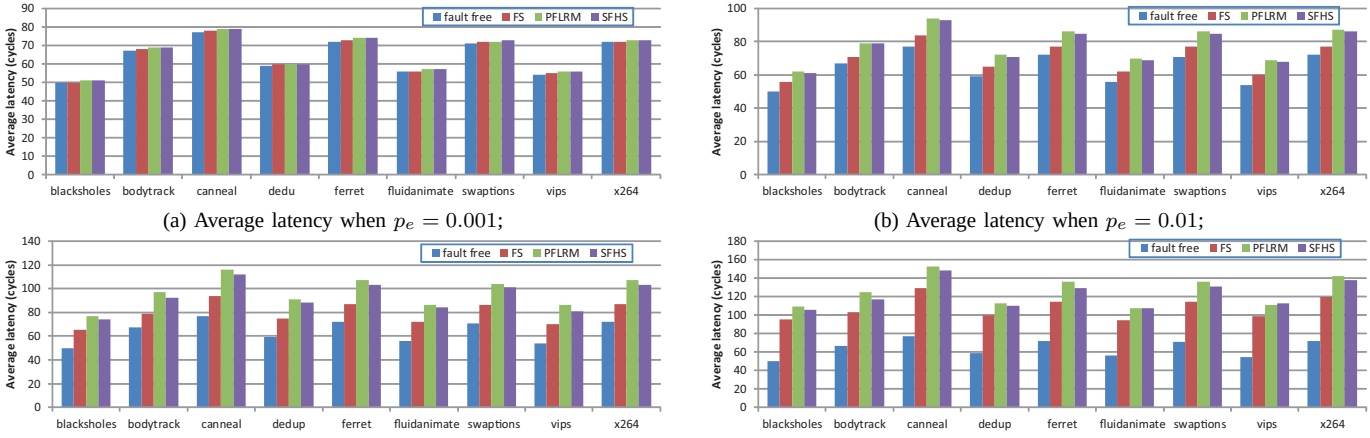
B. FS Performance with PARSEC

In this subsection, we evaluate our proposal with PARSEC benchmarks [8] traffic traces recorded with the Netrace [35] tool on the M5 full system simulator [36]. We replay the benchmark traces and inject the packets into our NoC platform according to the packet time flag while maintain the packets dependencies. When compared with the full system simulation, simulation with recorded traffic can better reflect the performance of the NoC system [21] as the performance fluctuations caused by the interaction between the cores and the NoC are removed. The packet length can be 4-flits and 20-flits according to the packet type. The transmission delay of each packet is counted since the packet is read out from the record in the source node till the tail flit is received by the destination node. The simulation results are illustrated in Fig. 10. We note that the links are divided into 8 sections for FS and SFHS.

We can observe that for all the three partially faulty link utilization strategies, the average packet transmission latency increases as the wire fault rate becomes higher. When the wire fault rate is quite low, i.e., $p_e = 0.001$, only several defected links with low fault level exist in the NoC. Given that benchmarks' FIRs are much lower than the saturation FIR there is no obvious difference between the 3 partially faulty link utilization approaches. As the p_e increases, the advantage of our proposal becomes obvious. For example, the FS packet transmission latency is on average 13% and 12% lower than that of PFLRM and SFHS, respectively, when $p_e = 0.01$, but the FS advantage increases to 28% and 22% latency reduction, respectively, when $p_e = 0.1$.

C. System with Heavily Defected Links

In this section, we divide the links into 8 sections and examine the performance of the proposed defective link utilizations strategy at different link deactivation threshold. The considered thresholds are T5, T6, T7, and T8, i.e., a link is deactivated when the number of broken sections is equal with or larger than 5, 6, 7, or 8, respectively. In the experiments, we randomly select 1%, 5%, 10%, and 15% of the NoC links and inject 20% to 40% broken wires into them to create Heavily



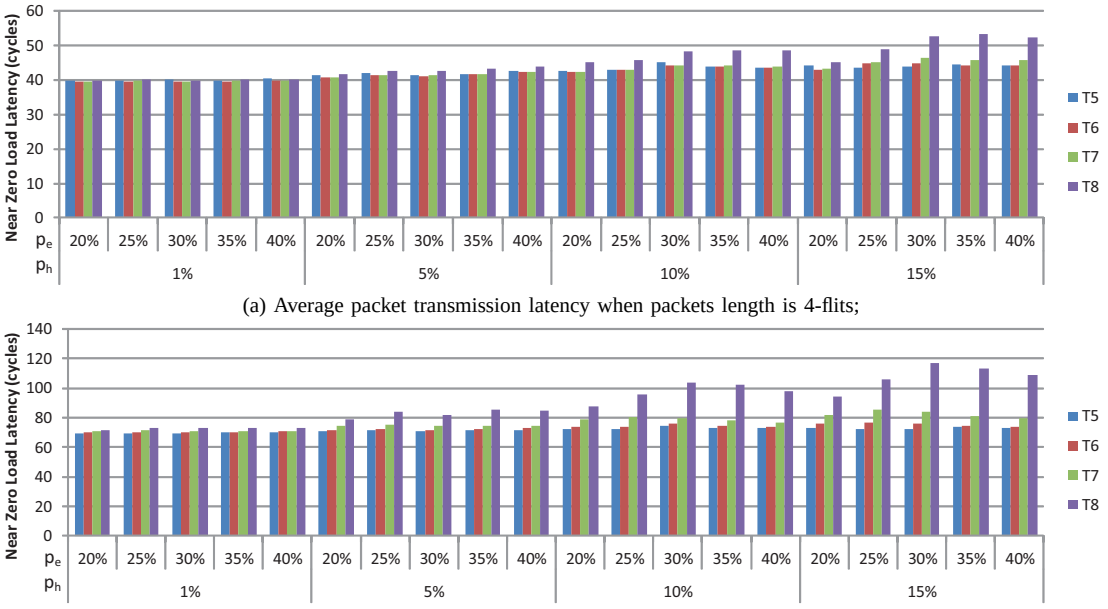
(a) Average latency when $p_e = 0.001$;

(b) Average latency when $p_e = 0.01$;

(c) Average latency when $p_e = 0.05$;

(d) Average latency when $p_e = 0.1$.

Fig. 10. Average packet transmission latencies of PARSEC Benchmarks at different fault rates. Links are divided into 8 sections for FS and SFHS.



(a) Average packet transmission latency when packets length is 4-flits;

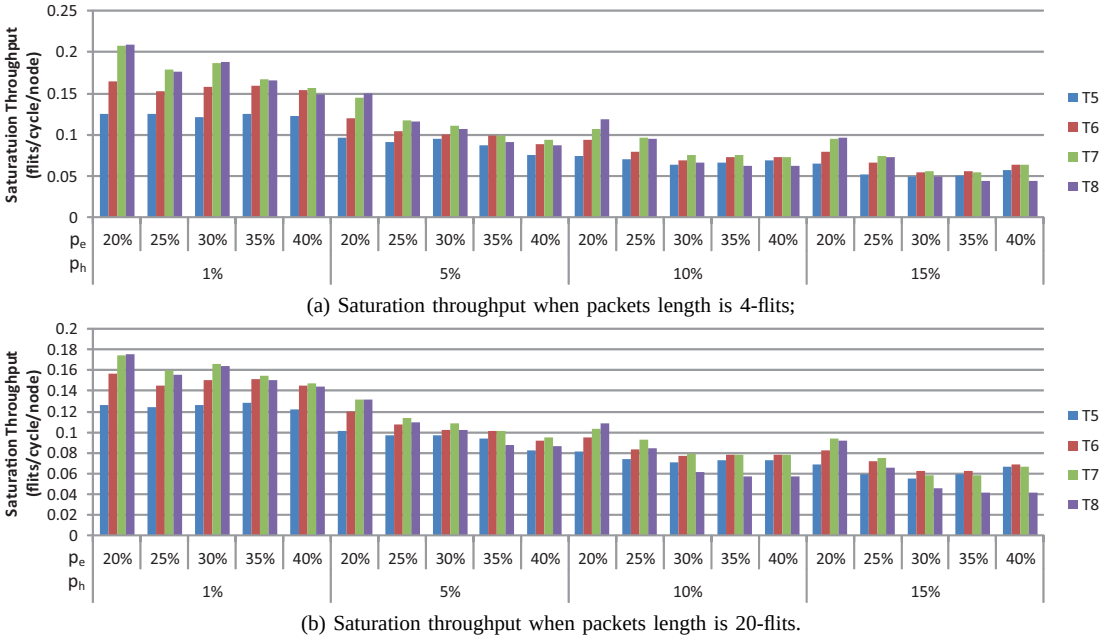
(b) Average packet transmission latency when packets length is 20-flits;

Fig. 12. The system average packet transmission latency when deactivate links with high fault level with different threshold.

Defected (HD) links, while the wire fault rate in the other links is 5%. The percentage of links with different number of broken sections at different wire fault rates are illustrated in Fig. 11. It indicates that most links have 5 or more broken sections when $p_e \geq 20\%$. As the link deactivation threshold increases from T5 to T7, an decreasing number of links are deactivated. However, as p_e grows, the number of broken sections in the HD links increases and thus more links are deactivated at the same link deactivation threshold.

The NoC system near zero load (FIR = 0.01 flits/cycle/node) packet transmission latency and saturation throughput at different wire fault rate configurations are illustrated in Fig. 12 and Fig. 13, respectively. The x-coordinates in the figures indicate the percentage of HD links (p_h) in the NoC and the percentage of faulty wires (p_e) in the HD links. We simulate the cases of short (4-flits) and long (20-flits) packets. Each experimental result is derived by averaging the results of 20 fault patterns with the same fault rate configuration. We note that when each link is divided into 8 sections, the average number of cycles required to successfully transmit a flit via a link with 5, 6, 7, and 8 broken link sections are 2.67, 4, 8, and ∞, respectively.

The results in Fig. 12 indicate that the average packet transmission latency at near zero traffic load in the T5, T6, and T7 cases has only small variation. Specifically, when compared with T6, the latency in T5 is slightly higher when the packet length is 4-flits but slightly lower when the packet length is 20-flits. This indicates that the link deactivation threshold for short packets should be set higher than that for long packets, which validates our analysis in Section IV-A. The difference between T6 and T5 is that the links that contain 5 broken sections are utilized in T6 but are deactivated in T5. According to the analysis in Section IV-A, the packet transmission latency via links at such fault level is similar with that via their misrouting-contour, thus T5 and T6 achieve similar near zero load performance. Due to the same reason, although slower links, i.e., the links with 6 broken sections, are still utilized in the T7 case, its near zero traffic load packet transmission latency is only slightly higher than that of T6, e.g., less than



(a) Saturation throughput when packets length is 4-flits;

(b) Saturation throughput when packets length is 20-flits.

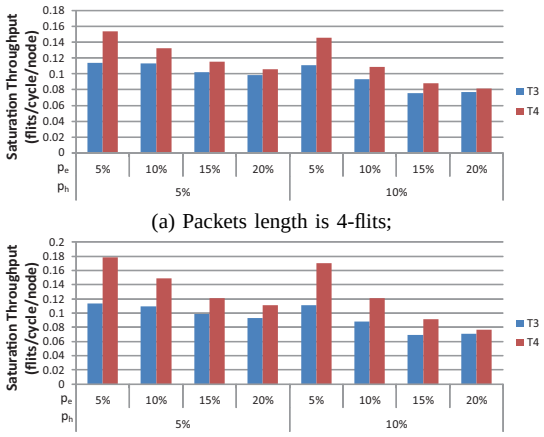
Fig. 13. The system saturation throughput when deactivate links with high fault level with different threshold.

9% even when $p_h = 15\%$. By comparison, links are utilized until all link sections are broken in the T8 case, case in which the links with a flit transmission latency of 8 cycles induce severe congestion in their upstream routers. Consequently the packet transmission latency in T8 is obviously higher than that of the cases with lower thresholds when $p_h \geq 5\%$.

The results in Fig. 13 indicate that T7 can achieve the highest saturation throughput for most of the fault rate configurations. We can also observe that as the link deactivation threshold increase from T5 to T7, although the near zero load packet transmission latency increases slowly, the saturation throughput is quickly improved. This is caused by the fact that when the NoC traffic load is high, deactivating HD links that contain 5 and 6 broken sections cause high congestion on their misrouting-contours, and thus it is more beneficial to directly transmit packets along these HD links. In the T8 case, the links that have 7 broken sections are so slow that the congestion in their upstream routers when they are utilized is much severer than the congestion in the misrouting-contours when they are deactivated. In the extreme case all VCs in an input port can be occupied by packets that are transmitted via such a slow TX link, case in which all subsequent packets are blocked even if they will be routed to other output ports. Consequently the saturation throughput at T8 is lower than that at T7.

It is worth to mention that when $p_h > 0.10$ in the NoC and $p_e > 30\%$ in the HD links, too many links are deactivated and some routers are also deactivated by UPF-FTRA to avoid deadlock. The number of deactivated routers increases as p_e grows. Consequently fewer packets are injected into the NoC and the near zero load packet transmission latency decreases and the saturation FIR for each node becomes higher.

In conclusion, when the NoC links are divided into 8 sections and partially broken links are utilized by means of the FS method, deactivating links that have 7 or more broken sections can efficiently balance the requirements for low near zero load packet transmission latency and high saturation throughput. In other words, the links should be utilized when their flit transmission latency is 4 cycles or less.



(a) Packets length is 4-flits;

(b) Packets length is 20-flits.

Fig. 14. The system saturation throughput at different link deactivation threshold when each link is split into 4 sections. A link is deactivated if 3 and 4 sections are broken in the T3 and T4 cases, respectively.

When the links are divided into less, e.g., 4, sections the average link fault level is higher at the same wire fault rate, thus if a link is deactivated the packet transmission latency on its misrouting-contour also becomes longer. This means that 4-section links should only be deactivated when the flit transmission latency on them is longer than 4 cycles, i.e., when all link sections are broken. This is also proved by the results illustrated in Fig. 14 that the saturation throughput for T4 is on average 33% and 18% higher than that at T3 for 4-flit and 20-flit packets, respectively.

D. Area and Power Cost

The area and power overheads of the four different link fault-tolerant methods, i.e., FS, PFLRM, SFHS, and spare wire replacement, are presented in Table. II. Our proposal and

TABLE II
POWER AND AREA OVERHEAD OF DIFFERENT LINK FAULT-TOLERANT METHODS

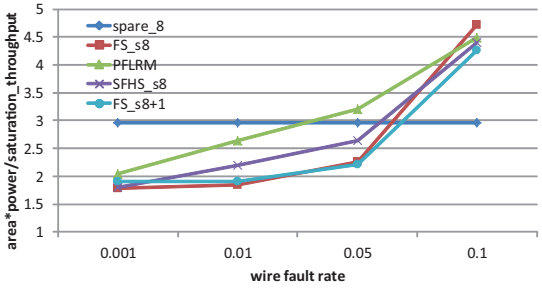
	Basic router	Spare wire		FS		PFLRM	SFHS		FS_extra_section		UPF_FTRA
		8 wires	4 wires	s8	s4		s8	s4	s8	s4	
Area	64813	55942	39727	27413	15812	15363	14407	7350	30827	17757	3040
(μm^2)	0%	86%	61%	42%	24%	24%	22%	11%	48%	27%	4.7%
Power	25.14	14.85	12.06	6.49	4.76	6.17	3.02	1.90	7.23	5.08	0.61
(mW)	0%	59%	48%	26%	19%	25%	12%	7.6%	29%	20%	2.4%

SFHS are evaluated with two versions containing 4 (s4) and 8 (s8) link sections, and the spare wire replacement method is evaluated with two versions containing 4 and 8 spare wires. From the Table we can observe that, the FS area and power overheads are lower than the ones of spare wire replacement, but higher than the ones of PFLRM and SFHS. For example, when compared with the baseline router, the FS_s8 area overhead is 42%, while those of 8 spare wires, PFLRM, and SFHS_s8 are 86%, 24%, and 22%, respectively; the FS_s8 power overhead is 26%, while those of 8 spare wires, PFLRM, and SFHS_s8 are 59%, 25%, and 12%, respectively. The FS_s4 area and power overheads also falls between those of 4 spare wires and SFHS_s4. It is worth to note that FS_s4 requires similar silicon area cost and less power consumption than PFLRM but provides better system performance when the wire broken rate is less than 0.01.

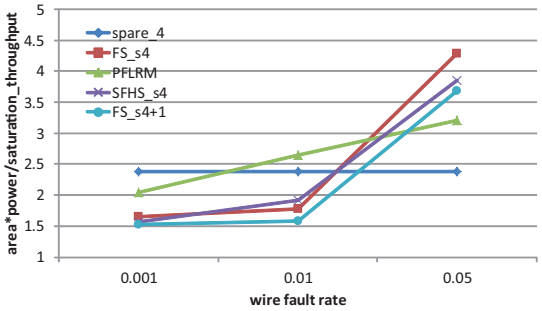
It is illustrated in Table II that adding one redundant link section to each link in the context of the FS method (FS+1) increases the area and power consumption by 6% and 3%, respectively, in the S8 case, and 3% and 1%, respectively, in the S4 case. We note that the number of wires is increased by 12.5% and 25% for the S8 and S4 cases, respectively.

When we further introduce the UPF-FTRA into the NoC system, another 4.7% area cost and 2.4% power consumption is required when compared with the basic router which does not tolerate any permanent faults.

To give a comprehensive overview on the implementation cost and performance of different link fault tolerant strategies, we compute their Area*Power/Saturation_throughput (AP/S) metric value and illustrate the normalized results, to that of the baseline router, in Fig. 15. A strategy that can achieve a low AP/S value, i.e., high saturation throughput and low area and power cost, is preferred. We can observe that when the wire fault rate (p_e) is as low as 0.001, FS, FS+1, and SFHS achieve the similar AP/S value, which is lower than that of PFLRM and spare wire replacement. When p_e is 0.01 and 0.05, FS_s8 and FS_s8+1 always achieve lower AP/S value than SFHS, PFLRM, and the spare wire replacement method. When p_e is as high as 0.1, which is unlikely to happen in practice, all partially faulty link utilization strategies induce high saturation throughput reduction and the spare wire replacement method becomes the most effective link fault tolerant strategy. Thus in practice FS is an effective method to tolerate faulty link wires and to utilize remained link bandwidth.



(a) With 8 spare wires or sections;



(b) With 4 spare wires or sections;

Fig. 15. Normalized value of area*power/saturation_throughput metric of different link fault tolerant strategies. Lower is better.

VI. CONCLUSIONS

In this paper, we proposed a Flit Serialization (FS) method to efficiently utilize partially defective links. The FS approach divides the links into a number of equal width sections, and serializes sections of adjacent flits to transmit them on all fault-free link sections to mitigate the unbalance between the flit size and the actual link bandwidth. We also proposed the link augmentation with one redundant section as a low cost mechanism to further increase the link dependability. To diminish congestion caused by Heavily Defected (HD) links, we discussed the optimal link deactivation threshold by comparing the zero load packet transmission latency on the HD links and that on the shortest alternative path. Our proposal is evaluated with synthetic traffic and PARSEC benchmarks. Experimental results indicate that the FS method can achieve lower area*power/saturation_throughput value than all state of the art link fault tolerant strategies. With a redundant section in each link, the NoC saturation throughput can be largely improved than just utilizing FS, e.g. 18% when 10% of the NoC wires are broken. Simulation results we obtained at various wire broken rate configurations indicate that we achieve the highest saturation throughput if 4- or 8-section links with a flit transmission latency longer than 4 cycles are deactivated.

REFERENCES

[1] G. Blake, R. G. Dreslinski, and T. Mudge, "A survey of multicore processors," *IEEE Signal Process. Mag.*, vol. 26, pp. 26–37, Nov. 2009.
[2] T. Dumitras, S. Kerner, and R. Marculescu, "Towards on-chip fault-tolerant communication," in *Proc. Asia and South Pacific Design Automation Conference*, Jan. 2003, pp. 225–232.
[3] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys*, vol. 38, pp. 1–51, Mar. 2006.

[4] C. Grecu, A. Ivanov, R. Saleh, and P. P. Pande, "Noc interconnect yield improvement using crosspoint redundancy," in *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, Oct. 2006, pp. 457–465.

[5] O. Unsal *et al.*, "Impact of parameter variations on circuits and microarchitecture," *IEEE Micro*, vol. 26, no. 6, pp. 30–39, Nov./Dec. 2006.

[6] C. Hernandez, F. Silla, V. Santonja, and J. Duato, "A new mechanism to deal with process variability in noc links," in *Proc. IEEE International Symposium on Parallel and Distributed Processing*, May 2009, pp. 1–11.

[7] S. Borkar, "Designing reliable systems from unreliable components: The challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, Nov./Dec. 2005.

[8] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, Jan. 2011.

[9] T. Lehtonen, D. Wolpert, P. Liljeberg, J. Plosila, and P. Ampadu, "Self-adaptive system for addressing permanent errors in on-chip interconnects," *IEEE Trans. VLSI Syst.*, vol. 18, no. 4, pp. 527–540, Apr. 2010.

[10] A. Vitkovskiy, V. Soteriou, and C. Nicopoulos, "A fine-grained link-level fault-tolerant mechanism for networks-on-chip," in *Proc. IEEE International Conference on Computer Design*, Oct. 2010, pp. 447–454.

[11] M. Palesi, S. Kumar, and V. Catania, "Leveraging partially faulty links usage for enhancing yield and performance in network-on-chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 3, pp. 426–440, Mar. 2010.

[12] T. Lehtonen, P. Liljeberg, and J. Plosila, "Online reconfigurable self-timed links for fault tolerant noc," *VLSI Design*, vol. 2007, pp. 1–14, 2007.

[13] Q. Yu and P. Ampadu, "Transient and permanent error co-management method for reliable networks-on-chip," in *Proc. IEEE/ACM International Symposium on Networks on Chip*, Jul. 2010, pp. 52–59.

[14] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, J. Plosila, and P. Liljeberg, "Optimized q-learning model for distributing traffic in on-chip networks," in *Proc. IEEE International Conference on Networked Embedded Systems for Every Application*, Dec. 2012, pp. 1–8.

[15] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip," *IEEE Trans. Comput.*, vol. 57, no. 6, pp. 809–820, Jun. 2008.

[16] M. Ebrahimi *et al.*, "Haraq: Congestion-aware learning model for highly adaptive routing algorithm in on-chip networks," in *Proc. IEEE/ACM International Symposium on Networks on Chip*, May 2013, pp. 19–26.

[17] M. Palesi, S. Kumar, and V. Catania, "Bandwidth-aware routing algorithms for networks-on-chip platforms," *IET Computers and Digital Techniques*, vol. 3, no. 5, pp. 413–429, Sep. 2009.

[18] L. Schwiebert and D. N. Jayasimha, "Optimal fully adaptive wormhole routing for meshes," in *Proc. Supercomputing*, Nov. 1993, pp. 782–791.

[19] C. Grecu, A. Ivanov, R. Saleh, and P. P. Pande, "Testing network-on-chip communication fabrics," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 12, pp. 2201–2213, Dec. 2007.

[20] E. Salminen, A. Kulmala, and T. Hamalainen, "Survey of network-on-chip proposals," White Paper, Open Core Protocol International Partnership (OCP-IP), 2008.

[21] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann Publishers Inc., 2003.

[22] W. Tsai, D. Zhen, S. Chen, and Y. Hu, "A fault-tolerant noc scheme using bidirectional channel," in *Proc. Design Automation Conference*, Jun. 2011, pp. 918–923.

[23] R. Tamhankar *et al.*, "Timing-error-tolerant network-on-chip design methodology," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 7, pp. 1297–1309, Jul. 2007.

[24] C. Chen and S. Cotofana, "Towards an effective utilization of partially defected interconnections in 2d mesh nocs," in *Proc. International Symposium on VLSI*, Jul. 2014, pp. 492–497.

[25] M. Ebrahimi, M. Daneshtalab, J. Plosila, and H. Tenhunen, "Minimal-path fault-tolerant approach using connection-retaining structure in networks-on-chip," in *Proc. IEEE/ACM International Symposium on Networks on Chip*, Apr. 2013, pp. 1–8.

[26] C. Glass and L. Ni, "Fault-tolerant wormhole routing in meshes without virtual channels," *IEEE Trans. Parallel Distrib. Syst.*, vol. 7, no. 8, pp. 620–636, Jun. 1996.

[27] S. Chalasani and R. V. Boppana, "Communication in multicomputers with nonconvex faults," *IEEE Trans. Comput.*, vol. 46, no. 5, pp. 616–622, May 1997.

[28] C. Chen and G. Chiu, "A fault-tolerant routing scheme for meshes with nonconvex faults," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 5, pp. 467–475, May 2001.

[29] C. Chen and S. Cotofana, "An effective routing algorithm to avoid unnecessary link abandon in 2d mesh noc," in *Proc. Euromicro Conference on Digital System Design*, Sep. 2013, pp. 374–379.

[30] S. Kim and T. Han, "Fault-tolerant wormhole routing in mesh with overlapped solid fault regions," *Parallel Computing*, vol. 23, no. 13, pp. 1937–1962, Dec. 1997.

[31] K. Aisopos, A. DeOrio, L. Peh, and V. Bertacco, "Ariadne: Agnostic reconfiguration in a disconnected network environment," in *Proc. International Conference on Parallel Architectures and Compilation Techniques*, Oct. 2011, pp. 298–309.

[32] F. Chaix, D. Avresky, N. Zergainoh, and M. Nicolaidis, "Fault-tolerant deadlock-free adaptive routing for any set of link and node failures in multi-cores systems," in *Proc. IEEE International Symposium on Network Computing and Applications*, Jul. 2010, pp. 52–59.

[33] V. Puente, J. Gregorio, F. Vallejo, and R. Beivide, "Immunet: Dependable routing for interconnection networks with arbitrary topology," *IEEE Trans. Comput.*, vol. 57, no. 12, pp. 1676–1689, Dec. 2008.

[34] Y. Lu, J. McCanny, and S. Sezer, "Exploring virtual-channel architecture in fpga based networks-on-chip," in *Proc. IEEE International SOC Conference*, Sep. 2011, pp. 302–307.

[35] J. Hestness and S. W. Keckler, "Netrace: Dependency-tracking traces for efficient network-on-chip experimentation," Department of Computer Science, The University of Texas at Austin, Austin, Texas, Tech. Rep. TR-10-11, May 2010.

[36] N. Binkert, R. Dreslinski, L. Hsu, K. Lim, A. Saidi, and S. Reinhardt, "The m5 simulator: Modeling networked systems," *IEEE Micro*, vol. 26, no. 4, pp. 52–60, Jun./Aug. 2006.

Changlin Chen Biography text here.

Yaowen Fu Biography text here.

Sorin Cotofana Biography text here.