2013 16th Euromicro Conference on Digital System Design

# An Effective Routing Algorithm to Avoid Unnecessary Link Abandon in 2D Mesh NoCs

Changlin Chen, Sorin D. Cotofana
Computer Engineering, Software and Computer Technology
Delft University of Technology, Delft, the Netherlands
Email: {c.chen-2, S.D.Cotofana}@tudelft.nl

***Abstract*—In NoCs where each interconnection between neighboring routers is composed of a pair of unidirectional links, a broken link usually leads to the abandon of the entire interconnection, even if the other one is still functional. In this paper, we propose a fault tolerant Routing Algorithm (RA) which can efficiently utilize these fault free links when their pair broken links have available misrouting-contour sides. Constraints on the usage of Virtual Channels are adaptively applied according to the fault distribution, to avoid deadlock and unnecessary resource reservation. When compared with solid fault region tolerant RAs, which always abandon the entire interconnection, the proposed algorithm has twice higher saturation point under synthetic uniform traffics, and can on average diminish the execution time overhead for the evaluated applications, *sample* and *satell*, by 62.6% and 76.6%, respectively. Our experiments indicate that the embedding of the proposed algorithm into a baseline router increases the area cost and power consumption by 7.43% and 4.43%, respectively, which is not that significant given that the platform area is usually dominated by the computing cores area.**

***Keywords*-Networks-on-Chip; Fault tolerant; Routing algorithm; Unidirectional links;**

## I. Introduction

The keeping on shrinking of transistor dimensions enables an increasing number of cores to be integrated on a single chip. Networks-on-Chip (NoC) are utilized to enable high performance inter-core communication. However, smaller transistor size also increases the possibility of defects in chip components, especially in long distance NoC links. Being prone to failures caused by manufacturing defects [1], chip wear out effects [2], and Process Parameter Variations (PPV) [3], links can contains broken wires or even be totally out of service.

To enable NoC performance graceful degradation, links with different fault degrees should be treated differently. Links with a small number of faulty wires should better be utilized relying on a Partially Faulty Link Usage Method (PFLUM), e.g., [4], [5], while links that have too many broken wires or are totally out of service should be discarded and make use of Routing Algorithms (RA) that can route the packets along alternative paths.

Conventionally, each interconnection between adjacent NoC routers is composed of a pair of unidirectional links, each of them having its own flow control wires and handling either outgoing or incoming traffic. [6] suggests to replace the unidirectional links with bidirectional ones. When one link is broken, the other one is utilized in both directions and results in half-duplex communication. However, unidirectional links are still attractive as it is simpler to implement their control logic and to address timing error issues [7]. In this paper, we focus on NoC architectures that utilize unidirectional links.

Assuming links have the same fault rate and are physically independent from each other, the probability that both links in an interconnection are broken is typically low, especially when PFLUM is used [4]. Thus, if the unpaired functional (UPF) links were utilized rather than discarded, the system performance degradation can be reduced. This creates the demands for RAs that can efficiently utilize these links.

Numerous Fault Tolerant (FT) RAs have been proposed for 2D mesh networks, e.g., [8]–[19]. Many of them treat the two links in one interconnection as an entire to guarantee deadlock free routing, e.g., [8]–[14]. Other RAs that have no such constraint usually need complicated path search algorithms or routing tables, which induce high silicon cost in NoC systems.

In this paper, we propose a One-Faulty-Link Tolerant (OFLT) RA that can efficiently utilize UPF links. Generally speaking, it is hard for a router to decide whether UPF links should be used to transmit packets, as the packets may be forced to return to it. An equivalent question is whether partially nonfunctional interconnections should be discarded without considering the existence of UPF links. In OFLT, a router makes the decision by checking if packets can be routed to a neighboring router along a shortest alternative path around the broken link. If true, UPF links are utilized with baseline RA and broken links are tolerated with the proposed OFLT algorithm. Otherwise, a fault region is formed and tolerated by a conventional Solid Fault Region Tolerant (SFRT) RA [13]. Deadlock is avoided by dynamically restricting the Virtual Channel (VC) usage according to the fault distribution.

Our analysis and experimental results indicate that the proposed OFLT algorithm has the following advantages:

1) UPF links in defective interconnections are efficiently utilized rather than being wasted.
2) VC usage constraint is adaptively applied to avoid unnecessary resource reservation.
3) The OFLT saturation points are on average twice higher than the ones of state of the art SFRT RAs, which entirely abandon UPF links.

The rest of the paper is organized as follows. Section II presents a brief survey of related work. Section III describes the proposed OFLT routing algorithm. Section IV analyzes the OFLT theoretical performance improvement. Section V presents evaluation results while Section VI concludes the presentation.

## II. Related Work

Numerous RAs have been proposed to tolerate broken links or routers in NoC systems. Based on whether the number and fault patterns are bounded, they can be classified into three main groups [19].

The first group, e.g., [8]–[10], can tolerate a bounded number of faults by modifying the turning rules of baseline RAs without causing deadlock. The RAs are mainly turn based and do not need VCs. The system has minimal performance loss when less than a bounded number faults occur but ends up in a deadlock otherwise. These algorithms can be modified to utilize resilient UPF links. However, the bounded fault number restricts their application scope.

The second RA group, e.g., [11]–[14], can tolerate an unbounded number of faults but requires the fault regions to have solid shapes. Otherwise, some fault free routers have to be deactivated to make the shape solid. Packet turning rules or VC usage constraints are only changed around fault regions. In the fault free region, the baseline RA is utilized. On the boundaries of fault regions, packets are routed clockwise or anticlockwise in reserved VCs to avoid deadlock. RAs of this group have the advantage that can cooperate with any baseline RAs. However, the usage of VCs are tightly restricted thus the system performance reduces substantially in the presence of any kind of faults. Being restricted by the solid fault region requirements, these RAs cannot make use of the UPF links.

RAs of the third group, e.g., [15]–[19], can tolerate an unbounded number of faults and unconstrained fault patterns. A node can be reached as long as it is connected to the NoC. In [15], [16], a packet is transmitted following a baseline RA before it is blocked by a faulty link or router. Otherwise, when this happens, the packet tries all potential output ports and records the successful selections in the head flit. The final successful transmission path is used to indicate how the following packets should be transmitted. This requires to change packet length dynamically or to reserve data bits to store the transmission choices. To avoid deadlock when packets break the baseline turning rules, VCs are usually statically reserved as escape channels. Differently, [17], [18] search available routing paths by doing graphic search during NoC reconfiguration when a new permanent fault is detected. A routing table is maintained in each router to indicate the output ports to requested destinations. For both path searching methods, the length of the routing path and the routing table size are proportional to the NoC size. The Vicis method proposed by [19] find the most suitable turn rules in each router offline. It can theoretically tolerate all kinds of faults but does not guarantee deadlock freeness. Updating the new turn rules in each router also introduces extra costs.

Although RAs of the third group have the potential to utilize UPF links, they have their own drawbacks when used in wormhole switched NoCs. NoC routers usually utilize a certain number of VCs and replicate the routing units in each port [20]. VCs are used to solve the Head-of-Line (HOL) [22] issues to improve throughput [21]. Although chip reliability is decreasing nowadays, the permanent fault rate is typically low in the normal use period of a chip's lifetime [23]. Reserving VCs statically for fault tolerant purpose is a waste of resource when the NoC has no broken links. Moreover, links with low fault degree can still be utilized by PFLUM. Routing unit in each port can calculate routing results for received packets immediately even if packets arrive at each port simultaneously. Logic based distributed routing mechanism is usually preferred to routing tables to save NoC silicon cost.

In view of the previous discussion, we propose a low cost One-Faulty-Link Tolerant RA in this paper. The number of faults is unlimited as long as a basic condition (stated in Section III) is satisfied. Our approach enables graceful system performance degradation by efficiently utilizing UPF links. Although VCs are required to avoid deadlock, the VCs usage is adaptively restricted around broken links to avoid unnecessary resource reservation. OFLT is logic based and does not require modification to packet formats. Around the clustered faults where the basic condition is not satisfied, a conventional SFRT RA, e.g., [13], can be utilized to tolerate the faults as a fault region. Because OFLT and SFRT have similar mechanisms to detect and store the fault distribution, implementing both RAs in one router does not induce significant silicon overhead.

## III. One-faulty-link Tolerant Routing

When one link in an interconnection is broken while the other one is still functional, only messages that have the same direction with the broken link are blocked in the transmitter side router. Data flow with opposite direction can continue on the functional link. We note here that messages are transmitted in the form of packets and that the two terms, message and packet, are used indiscriminately in this paper.

### A. Preliminaries

In a 2D mesh NoC, we consider that each router, which is not on an edge, is neighboring with 24 links and 8 routers as is illustrated in Fig. 1. We embed a 24-bit *link_status_register* in each router to store the neighboring links' statuses. The relative register bit of a link is illustrated in Fig. 1 as seen by router C. A register bit is set to '0' if its relative link is broken and '1' if the link is functional. All links incident to a faulty or deactivated router are marked as faulty.

Each router $n$ is represented by its position in the NoC, $n{=}(x, y)$. Adjacent routers $n_0$ and $n_1$ are connected by two unidirectional links $L_{01}{=}{<}n_0, n_1{>}{=}{<}(x_0, y_0), (x_1, y_1){>}$ and $L_{10}{=}{<}n_1, n_0{>}{=}{<}(x_1, y_1), (x_0, y_0){>}$. The directions of links ${<}(x, y), (x{+}1, y){>}$, ${<}(x, y), (x{-}1, y){>}$, ${<}(x, y), (x, y{-}1){>}$, and ${<}(x, y), (x, y + 1){>}$ are WE, EW, SN, and NS, respectively.

**Definition 1.** *The one-faulty-link situation occurs when only one link between two neighboring routers is faulty, while the other one is functional.*

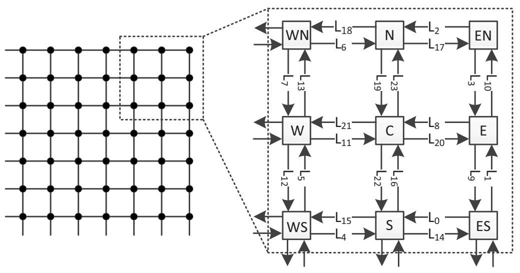


Fig. 1. The link position in the C router *link_status_register*.

Note that we use the one-faulty-link definition to illustrate the defective status of an interconnection between two adjacent routers. At a certain moment in time, depending on the fault conditions, numerous such one-faulty-link interconnections may be present in the entire NoC.

When a link is broken, it is of interest to know if there are shortest alternative paths around that link, and for this, we employ the concept of *misrouting-contour*.

**Definition 2.** *The* misrouting-contour *of a link L is composed of the neighboring links that can be utilized to create an alternative path to route a packet from L's source to L's destination.*

For example, the misrouting-contour of $L_{20}$ in Fig. 1 is composed of $L_{23}$, $L_{17}$, $L_3$, $L_{22}$, $L_{14}$, and $L_1$. While the misrouting-contour of $L_1$, which locates on an edge of the NoC, is composed of fewer links, i.e., $L_0$, $L_{16}$, and $L_{20}$.

Based on the outgoing direction of a link, its misrouting-contour can be divided into left and right sides when the link is not on an edge. For example, the misrouting-contour of $L_8$ can be divided into the left side, including $L_{10}$, $L_2$, and $L_{19}$, and the right side, including $L_9$, $L_0$, and $L_{16}$.

## B. Failure Diagnosis and OFLT Basic Condition

We assume that a newly broken link is detected when the number of errors in flits transmitted on the link exceeds a threshold in a fixed period, or the number of broken wires in the link is too big. A faulty router can be detected by its neighboring routers [15]. An active router is aware of the health status of all the links incident to it and shares the links' status information with adjacent routers. We propose to transmit the information using one serial signal *r_data* instead of the outgoing link because the link might be broken. Since only one serial line is required, it can be protected by Triple Modular Redundancy (TMR) method with marginal overhead. The format of the data transmitted on *r_data* in each direction is illustrated in Table I. In the table, *1010* is the synchronization sequence, which is transmitted first, *safe/unsafe* is the status of the current router, *X_TX/RX* (*X* is N, S, W, or E) is the status of the TX/RX link in direction *X*. After receiving the information from *r_data*, a router is aware of the status of all its 24 neighboring links. If it detects a faulty neighboring router, all the links incident to that router are marked as faulty in the *link_status_register*.

TABLE I
FORMAT OF DATA TRANSMITTED ON R_DATA

| To | D0-D3 | D4 | D5 | D6 | D7 | D8 |
|---|---|---|---|---|---|---|
| East | 1010 | safe/unsafe | N_TX | N_RX | S_TX | S_RX |
| West | 1010 | safe/unsafe | N_TX | N_RX | S_TX | S_RX |
| South | 1010 | safe/unsafe | E_TX | E_RX | W_TX | W_RX |
| North | 1010 | safe/unsafe | E_TX | E_RX | W_TX | W_RX |

After the *link_status_register* is updated, each router decides whether the UPF links incident to it should be discarded or not. If at least one misrouting-contour side of a broken link is functional, the relative UPF link can be utilized and the broken link can be tolerated by the proposed OFLT algorithm. The at least one functional misrouting-contour side requirement is the **basic condition** to use OFLT.

When the basic condition is not satisfied for a broken link, the router at the transmitter side claims itself as *unsafe* and mark all UPF links incident to it as broken. The *unsafe* announcement and new links' status information are sent to neighboring routers via *r_data*. A router that receives the *unsafe* signal also claims itself as *unsafe* if there are faulty links incident to it. Again the router deactivates all UPF links and spreads the updated information to its neighbors. A router that received the *unsafe* signal but has no broken links incident to it still claims itself as *safe* and terminates the signal spread. Eventually the fault pattern is validated. All the routers that have generated or received the *unsafe* signal check if they are on the boundary of the faulty region. If true, packets will be transmitted in these routers following a conventional SFRT routing algorithm, e.g., [13]. All the routers in Fig. 2 are on fault region boundary. This procedure finishes in a finite number of cycles.

As illustrated in Table. II, the probability that the basic condition is not satisfied is rather low. Moreover, when links with low fault degree are utilized by means of PFLUM, the probability is even lower. The fault situations that impede the OFLT utilization are illustrated in Fig. 2. Combinations of two or more of them are also possible. Occurrences of these fault patterns usually implies that the fault rate in this area is high, thus it is reasonable to tolerate them as a fault region.

The main OFLT overhead lies on the detection and storage of fault distribution, which is also needed in SFRT RAs, while the routing logic is rather simple. Thus implementing both OFLT and a SFRT RA does not induce significant overhead.

After the fault pattern is established, the packet transmission can be resumed. If the head flit of a packet has not been transmitted before the fault happened, the packet will be transmitted on an alternative path after recovery. If the fault occurred after the head flit was transmitted but the tail flit is still in the upstream router, the flow control method proposed in [26] can be utilized.

## C. One-Faulty-Link Tolerant Routing Algorithm

For the sake of simplicity, we use the well know e-cube routing algorithm, such as XY, as underlying RA. We note here that other algorithms, e.g., west-first, can also be used

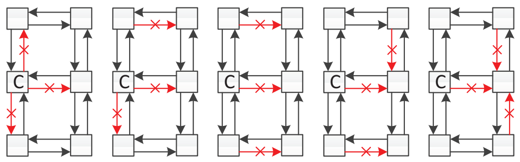


Fig. 2. Fault distributions for which the basic condition is not satisfied.

with simple modifications. With an e-cube routing algorithm, messages are first routed along one dimension and then on the other one in a 2D mesh NoC. The following description is based on the assumption that the basic condition is satisfied. Otherwise, OFLT degenerates to a conventional SFRT routing algorithm as discussed in the previous subsection.

When a message is generated at a source node $n_s = (x_s, y_s)$ and destined for node $n_d = (x_d, y_d)$, it is labeled as EW if $x_d < x_s$ or WE if $x_d < x_s$. When a message reaches its destination column, the message type changes to NS if $y_d > y_s$ or SN otherwise and cannot change again.

Messages are routed following the e-cube RA if the hop defined by the RA (e-cube hop) is not blocked. Otherwise, the message is misrouted along one side of the misrouting-contour of the faulty link. If the link is not on a NoC edge and both misrouting-contour sides are available, the message can be misrouted along the left or right side according to the position of its destination node relative to the faulty link. If the current node and the destination node are in the same row or column, either misrouting-contour side can be chosen.

If both links between two neighboring routers are broken, although contours of the two links are overlapped, their misrouting-contours are composed of totally different links. For example, as depicted in Fig. 1, the links included in the misrouting-contour of $L_8$ are $L_{10}$, $L_2$, $L_{19}$, $L_9$, $L_0$, and $L_{16}$, while the links included in the misrouting-contour of $L_{20}$ are $L_{22}$, $L_{14}$, $L_1$, $L_{23}$, $L_{17}$, and $L_3$. Thus the proposed OFLT algorithm is still applicable in this situation.

We use 4 VCs, labeled as $C_0$, $C_1$, $C_2$, and $C_3$, in each router port. Four VCs are widely used in NoC proposals [24] and is proved that they can efficiently improve the NoC throughput [21]. The VCs are freely used by any message type in the fault free region but dynamically constrained in the misrouting-contours of faulty links. In a misrouting-contour, one VC is reserved for messages that have the same direction as the faulty link. For example, $C_0$ is reserved for WE messages in the misrouting-contour of a WE faulty link, $C_1$ is reserved for EW messages in the misrouting-contour of an EW faulty link, $C_2$ is reserved for NS messages in the misrouting-contour of a NS faulty link, and $C_3$ is reserved for SN messages in the misrouting-contour of a SN faulty link. The unreserved VCs are shared by other types of messages. For example, in the misrouting-contour of a WE faulty link, a WE message can only use $C_0$ while other types of messages can apply for $C_1$, $C_2$, and $C_3$ . In the case that misrouting-contours of two or more faulty links are overlapped, a message uses the reserved VC if its direction is the same with one faulty link, or apply for an unreserved VC otherwise. This enables the VC usage constraint adaptive application to ensures deadlock freeness. However, because only one VC is available for each message type in a misrouting-contour, HOL blocking can be easily formed when the VC is requested by multiple packets. We label this VC usage strategy as *tight VC usage constraint*.

When the basic condition is satisfied, Lemma 1 is true.

**Lemma 1.** *An EW (WE) message will never use a link with WE (EW) direction, and a NS (SN) message will never use a link with SN (NS) direction.*

*Proof:* An EW (WE) message is always first transmitted in a row to the West (East). When it is blocked by a faulty link before it reaches its destination column, according to the algorithm, it makes one hop to the north or south neighboring router and then turn West (East) again. For an EW (WE) message to be transmitted on a WE (EW) link, it must have made an $180^o$ turn. However, this is not possible because it is in conflict with the RA. If the message type is SN (NS), it is transmitted in its destination column to the North (South). When it is blocked by a faulty link, it leaves the column temporarily and then turn back to the destination column in three hops along the misrouting-contour of the faulty link. Thus NS (SN) links are not used in the misrouting. ■

In other words, one link will never be involved in the misrouting-contour of a faulty link that has opposite direction than it. Thus, according to our VC allocation rules, $C_0$ will never be reserved in an EW link. Accordingly, $C_1$, $C_2$, $C_3$ will never be reserved in WE, SN, and NS links, respectively. Even in the extreme case that a link is involved in misrouting-contours of three faulty links with different directions, there will be one VC not reserved by any message type. That VC can be shared by other types of messages. Thus, in the misrouting-contour of a faulty link, $C_0$ can be shared by EW, NS, and SN messages in EW links, $C_1$ can be shared by WE, NS, and SN messages in WE links, $C_2$ can be shared by EW, WE, and SN messages in SN links, and $C_3$ can be shared by EW, WE, and NS messages in NS links. In this way each packet can make use of two VCs, the reserved one and the shared one, at least. We label this VC usage constraint strategy as *loose VC usage constraint*.

For a link that locates on the boundary of a fault region, VC usage is decided by the conventional SFRT routing algorithm.

## *D. Deadlock and Livelock Freeness*

When the basic condition to use the proposed routing algorithm is satisfied, the following statements are true.

**Lemma 2.** *The VC usage only needs to be constrained in links that are involved in misrouting-contours. In other links, VCs can be selected according to the underlying RA.*

*Proof:* When a packet is blocked by a broken link, it is misrouted along the misrouting-contour of that link. Links that are not involved in the misrouting-contour are not used. In other words, these links are only utilized by normally routed packets. The turn rules to change a packet's transmission direction are decided by the underlying RA. There is no need to put extra constraints to the VCs usage in these links.

In misrouting-contours, forbidden turns in underlying RA are used to route packets around faulty links, leading to the possibility of deadlocks. Putting constraint to the VCs usage is a simple way to avoid deadlock. ■

**Lemma 3.** *The OFLT routing algorithm is deadlock free.*

*Proof:* We use $R_1$ and $R$ to represent the OFLT routing algorithm when tight and loose VC usage constraint strategies are applied, respectively. As the VC usage constraint is tighter in $R_1$ than that in $R$, $R_1$ is a routing subfunction of $R$. The Channel Dependency Graphs (CDG) of $R_1$ and $R$ are illustrated in Fig. 3.

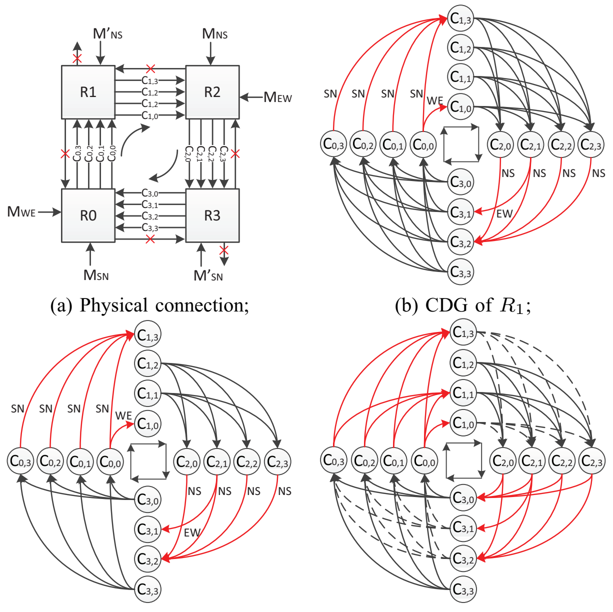
If deadlock exists, there must be clockwise or anticlockwise cyclic channel dependency. We use clockwise channel dependency as example to prove that loops cannot be formed in $R_1$. Anticlockwise channel dependency can be analyzed in the same way.

Fig. 3(a) depicts a typical situation when forbidden turns (N-E and S-W) in XY routing are used. The N-E turn happens when WE or SN messages are misrouted, and the S-W turn happens when EW or NS messages are misrouted. In wormhole switched NoCs, one packet can hold multiple routers. In Fig. 3(a), only the routers where packets make turns are illustrated.

In Fig. 3(b), new dependencies brought by $R_1$ are labeled with red color and the message types in which the dependencies could happen. The black arrowed arcs represent existing channel dependency in XY routing. A SN message ($M_{SN}$) can hold any of the four VCs in a SN link if the link is not in a misrouting-contour. Then $M_{SN}$ turns to a WE link when its normal hop is blocked. In the link, $C_{1,3}$ is reserved for $M_{SN}$ and cannot be used by another types of messages, because the link is involved in the misrouting-contour of the broken SN link incident to router R1. Similarly, a WE message ($M_{WE}$) can choose the left misrouting-contour side when it is blocked by a broken WE link. In the misrouting-contour, $C_{0,0}$ is reserved for $M_{WE}$ in the the SN link and $C_{1,0}$ is reserved for $M_{WE}$ in the WE link. Both $C_{0,0}$ and $C_{1,0}$ will not be used by any other types of messages. Same analysis applies to misrouted $M_{NS}$ and $M_{EW}$. Channel dependencies when W-N and E-S turns are used in misrouting ($M'_{SN}$ and $M'_{NS}$) are included in the CDG of underlying XY routing, they will not introduce new cycles in $R_1$. In conclusion, there is no clockwise cyclic channel dependency in $R_1$ as shown in Fig. 3(c). Similarly, there is no anticlockwise cyclic channel dependency in $R_1$ thus $R_1$ is deadlock free.

In the analysis, we did not consider how unreserved VCs are used. Thus their usage can be decided by underlying RA.

According to Theorem 3.1 in [22], $R$ is deadlock free because its routing subfunction $R_1$ is deadlock free. Dashed arcs in Fig. 3(d) do not exist when relative misrouting happens. Although the cyclic channel dependency $C_{0,*} \rightarrow C_{1,1} \rightarrow C_{2,*} \rightarrow C_{3,0}$ may be formed, each misrouted message has a reserved VC, which acts as the escape channel.



(a) Physical connection; (b) CDG of $R_1$;

(c) CDG of $R_1$ without loops; (d) CDG of $R$;

Fig. 3. Channel Dependency Graph of $R_1$ and $R$.

When the basic condition is not satisfied, OFLT degenerates to a conventional SFRT routing algorithm, which is already proved to be deadlock free. ■

**Lemma 4.** *OFLT routing algorithm is livelock free.*

*Proof:* When the basic condition is satisfied, blocked messages are always misrouted along misrouting-contours of broken links to downstream routers, which are closer to the destination. So the algorithm is also livelock free. ■

## IV. Performance Analysis

In this section, we analyze the efficiency of OFLT in utilizing UPF links and reducing communication overheads.

### A. Link Utilization Efficiency

We assume that each link has the same probability to be faulty and do Monte Carlo simulations to determine the average numbers of interconnections that contain broken links, interconnections in which both links are broken, and broken links that cannot satisfy the basic condition in an $8 \times 8$ 2D mesh NoC, when the link fault rate increases from 1% to 10%. The results are presented in Table II.

From Table II we can observe that the percentage of interconnections without any functional link is quite low. There are 224 unidirectional links in the NoC. Even when 10% of them are broken, only about 1.11 interconnections are totally broken, while 2.76 broken links have no fault free misrouting-contour, i.e., OFLT degenerates to a conventional SFRT routing algorithm around these faults. Although the faults cannot be distributed so evenly in a NoC system, we still can expect that most of the channels are only partially broken and tolerable by OFLT. Moreover, if PFLUM, e.g.,

TABLE II
AVERAGE NUMBERS OF INTERCONNECTIONS WITH DIFFERENT FAULT STATUSES IN A $8 \times 8$ 2D MESH

| link faulty rate | 1% | 2% | 3% | 4% | 5% | 6% | 7% | 8% | 9% | 10% |
|---|---|---|---|---|---|---|---|---|---|---|
| interconnections with broken links | 2.27 | 4.46 | 6.58 | 8.78 | 10.84 | 12.97 | 15.04 | 17.19 | 19.20 | 21.26 |
| interconnections with two broken links | 0.01 | 0.04 | 0.10 | 0.18 | 0.27 | 0.40 | 0.55 | 0.70 | 0.90 | 1.11 |
| broken links without misrouting-contour | 0.02 | 0.07 | 0.19 | 0.34 | 0.56 | 0.84 | 1.19 | 1.65 | 2.15 | 2.76 |

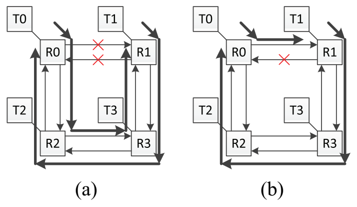


Fig. 4. Analysis of system performance improvement.

[4], [5], is applied, the number of broken links that cannot be tolerated by OFLT is even smaller.

Our analysis indicate that UPF links can be utilized by OFLT most of the time. Actually, when too many broken links cannot be tolerated by OFLT, the link fault rate must be already very high and the system performance will be severely degraded even if more complicated FT RAs were used. In such fault conditions, the chip would be replaced as most systems with high availability are actively maintained [20].

## B. Communication Overhead Reduction

Previous research suggests that in a well mapped NoC system, most data flows have low number of hops [27]. Assuming for example that two tasks, T0 and T1, are mapped onto two adjacent processor nodes (see Fig. 4), only one hop is required for them to reach each other when both links are functional. If one unidirectional link was broken, the other link will also be discarded in conventional methods. Then each task needs at least 3 hops to reach the other one. In the proposed OFLT method, the healthy link is still utilized thus one data flow direction is not affected by the broken link, which results in a reduced communication overhead.

## V. EVALUATION

To put the implication of OFLT in a better practical prospective, we evaluate and compare it with tightly related FT RAs. To this end, we implement every RA in the context of an $8 \times 8$ 2D mesh NoC Platform [25] at RTL level by using Verilog HDL. The baseline router has 3 pipeline stages: Routing Computation, combined VC/Switch Allocation, and Switch Traversal. A Link Transversal stage is added between adjacent routers. Each router has 5 Physical Channels (PC), each PC is shared by 4 VCs, and the buffer in each VC is 4-flit deep.

## A. Synthetic Traffic

We first insert different fault patterns into the NoC platform and run synthetic traffics on it. The basic RA is XY and the traffic pattern is uniform, which are both most frequently used in NoCs evaluation experiments [22]. We compare the OFLT performance with that of the SFRT algorithm (CCFT) in [13], and of the RA in [10] which can tolerate one faulty channel (OFCT) in a NoC. In Fig. 5, L<(3, 3), (4, 3)> means that the unidirectional link from router $(3, 3)$ to $(4, 3)$ is broken, while C<(3,3),(4,3)> means both links between the two routers are broken. Unless otherwise noted, OFLT uses the loose VC usage constraint strategy.

To be fair, 4 VCs are used in each RA. Note that even if a RA requires less VCs, e.g., 3 in CCFT and 1 in OFCT, the extra VCs can reduce HOL effects thus improve the NoC performance. The extra VCs are freely used by any message.

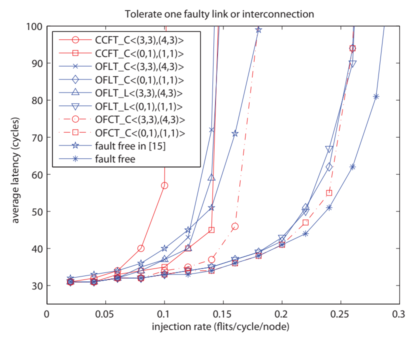
Fig. 5(a) presents the performances of different FT routing algorithms when one broken link or interconnection locates in different NoC positions. From the graphs we can observe that, faults that locate in the NoC center degrade system performance much severely than faults close to edges for any FT routing algorithm. When the fault locates close to NoC edges, the NoC performance when OFLT is utilized is only slightly worse than the fault free case. When the fault locates in the center, the saturation point, i.e., the packet injection rate for which the average transmission latency approaches infinity, of OFLT is 12.5% lower than that of OFCT. The reason is that OFCT does not constrain VC usage around a fault thus VCs are more freely used than in OFLT. In all evaluated one fault situations, OFLT substantially outperforms CCFT.

The performance of the RA proposed in [15] in the fault free case is also illustrated in Fig. 5(a). The algorithm divides the four VCs in each port into two classes and reserves each class for negative first or positive first routed messages. Simulation results prove that reserving VCs for fault tolerance degrade the system performance obviously even when no fault happens.

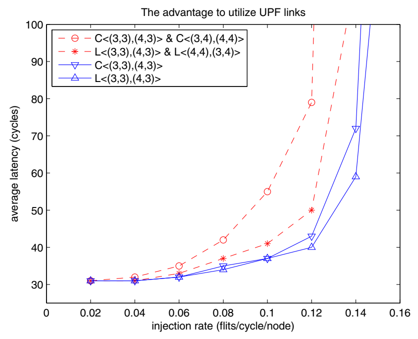
Fig. 5(b) indicates that it is beneficial to utilize the UPF links. Although the improvement is not much when only one unidirectional link is broken, it becomes more substantial when multiple such broken links exist.

The comparison between OFLT and CCFT when the NoC has different fault rates and faulty links are randomly generated is illustrated in Fig. 5(c). We can observe that OFLT has much higher performance than CCFT in all evaluated fault patterns. In particular, when the fault rate is 1%, 2%, 5%, and 10%, the OFLT saturation points are on average twice higher than the CCFT ones. The improvement is achieved by efficiently utilizing UPF links and by putting loose constraint to the VC usage around faults.

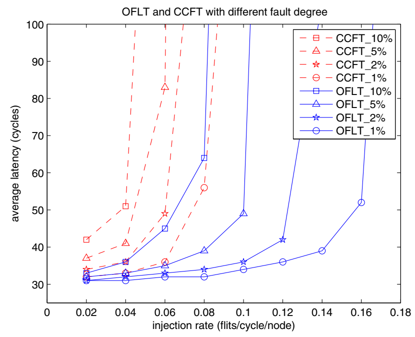
Fig. 5(d) presents the OFLT performance when tight and loose VC usage constraints are employed. When the tight constraint strategy is used, a misrouted packet can only apply for the VC reserved for it. HOL blocking can happen easily around a faulty link. When loose constraint strategy is used,



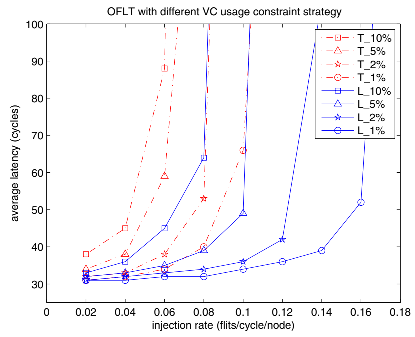
(a) Tolerate one faulty link or interconnection;



(b) The advantage to utilize UPF links;

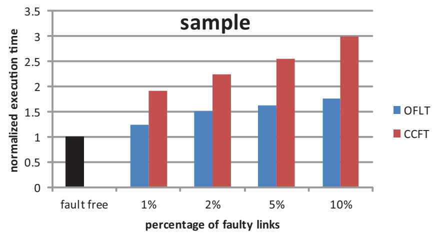


(c) OFLT and CCFT with different fault rate;

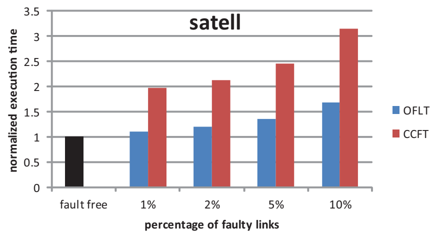


(d) OFLT with tight (T) and Loose (T) VC usage constraint;

Fig. 5. Performance of different fault tolerant routing algorithms.



(a) Application *sample*;



(b) Application *satell*;

Fig. 6. Normalized performance of applications when different percentages of faulty links exist.

each packet has at least two VCs to apply for, thus the congestion is partially solved. Consequently, OFLT achieves a better performance.

## B. *Recorded Traffic*

In this subsection, we evaluate OFLT with recorded traffic patterns derived from a realistic traffic benchmark suite, called MCSL [28]. Each traffic pattern is a trace of message transmission captured from a real application. Simulation results on recorded traffic can better reflect the performance of the NoC system [20]. Fault patterns resulting in different percentages of faulty links are generated randomly. We note that because the tasks in each application are statically mapped onto processing units, fault patterns that contain faulty or deactivated routers are not considered. The normalized execution time of the two considered applications in different fault circumstances is illustrated in Fig. 6.

Although the execution time of an application can be affected by multiple issues, including traffic load, fault pattern, routing algorithm, and so on, the performance of OFLT is much better than CCFT for both evaluated applications. In particular, OFLT can on average save by up to 62.6% and 76.6% of the execution time overheads required by CCFT, for *sample* and *satell*, respectively.

## C. *Area and Power Consumption*

Routers equipped with different RAs are synthesized (without optimization) using the Synopsys Design Compiler with TSMC 65-nm standard cell technology. The target frequency is 500MHz. The power consumption and area overhead corresponding to different FT RAs are presented in Table III.

We can observe that the overhead to implement the proposed

TABLE III
POWER AND AREA OVERHEAD OF DIFFERENT FAULT TOLERANT ROUTING ALGORITHMS

| Routing algorithm | Dynamic Power ($mW$) | Leakage Power ($\mu W$) | Area ($\mu m^2$) |
|---|---|---|---|
| Basic_XY | 14.66 / 0% | 378.2 / 0% | 55995 / 0% |
| OFLT | 15.31 / 4.43% | 407.2 / 7.67% | 60157 / 7.43% |
| CCFT | 15.21 / 3.75% | 403.2 /6.61% | 59611 / 6.46% |
| OFCT | 14.92 / 1.77% | 386.2 / 2.12% | 57295 / 2.32% |

algorithm is similar with that of CCFT. The area overhead for embedding OFLT into a baseline router is 7.43% of the one corresponding to the baseline router architecture and the dynamic power consumption increases by 4.43%. When comparing with OFCT, an RA which can tolerate a bounded number of faults and do not need any constraints to the usage of VCs, OFLT and CCFT require extra combinational logic and registers, besides the RC unit, to decide which VC can be granted to the just arrived packet. We note here that because routers represent a small percent of the silicon cost of the entire multi/many core system, this overhead to the router footprint does not result in a substantial system overhead.

## VI. Conclusion

In this paper, we proposed a One-Faulty-Link Tolerant (OFLT) routing algorithm that can efficiently utilize unpaired functional links in NoCs, when the basic condition, that their pair broken links have at least one misrouting-contour side, is satisfied. If the basic condition is not satisfied for a fault pattern, OFLT degenerates to a conventional Solid Fault Region Tolerant (SFRT) routing algorithm and tolerate the faults as a fault region. Usage of virtual channels is loosely constrained around faulty links to avoid deadlock. Experimental results demonstrate that, OFLT enables graceful system performance degradation when the link fault rate increases. The OFLT saturation points are on average twice higher than the ones of the SFRT algorithm in [13] when the NoC has different link fault degree. For the evaluated applications, *sample* and *satell*, OFLT can on average save by up to 62.6% and 76.6% of the execution time overheads required by CCFT. Embedding OFLT into a baseline router increases the area cost and power consumption by 7.43% and 4.43%, respectively.

## References

[1] C. Grecu, A. Ivanov, R. Saleh, and P. P. Pande, “NoC interconnect yield improvement using crosspoint redundancy,” IEEE DFT, pp. 457-465, Oct. 2006.

[2] S. Borkar, “Designing reliable systems from unreliable components: The challenges of transistor variability and degradation,” IEEE Micro, vol. 25, pp. 10-16, Nov.-Dec., 2005.

[3] O. Unsal, et al, “Impact of Parameter Variations on Circuits and Microarchitecture,” IEEE Micro, Vol. 26, pp.30-39, Nov.-Dec., 2006

[4] A. Vitkovskiy, V. Soteriou, and C. Nicopoulos, “A Dynamically Adjusting Gracefully Degrading Link-Level Fault-Tolerant Mechanism for NoCs,” IEEE Trans. CAD-ICS, pp.1235-1248, Aug. 2012.

[5] C. Chen, Y. Lu, and S. Cotofana, “A Novel Flit Serialization Strategy to Utilize Partially Faulty Links in Networks-on-Chip,” NOCS, pp. 124-131, May, 2012.

[6] W. Tsai, D. Zheng, S. Chen, and Y. Hu, “A Fault-Tolerant NoC Scheme Using Bidirectional Channel,” DAC, pp. 918-923, Jun. 2011.

[7] R. Tamhankar, et al., “Timing-Error-Tolerant Network-on-Chip Design Methodology,” IEEE Trans. CAD-ICS, pp. 1297-1310, vol.26, no. 7, Jul. 2007.

[8] C. Glass, L. Ni, “Fault-Tolerant Wormhole Routing in Meshes without Virtual Channels,” IEEE Trans. PDS, pp. 620-636, vol. 7, no. 6, Jun. 1996.

[9] Z. Zhang, A. Greiner, and S. Taktak, “A reconfigurabel routing algorithm for a fault tolerant 2D-mesh Network-on-Chip,” DAC, pp. 441-446, Jun. 2008.

[10] M. Valinataj, P. Liljeberg, and J. Plosila, “A fault-tolerant and hierarchical routing algorithm for NoC architectures,” NORCHIP, pp. 1-6, Nov. 2011.

[11] S. Chalasani, and R. V. Boppana, “Communication in multicomputers with nonconvex faults,” IEEE Trans. Computers, vol. 46, no. 5, May 1997.

[12] S. Kim, and T. Han, “Fault-tolerant wormhole routing in mesh with overlapped solid fault regions,” Parallel Computing, pp. 1937-1962, 1997.

[13] C. Chen, and G. Chiu, “A Fault-Tolerant Routing Scheme for Meshes with Nonconvex Faults,” IEEE Trans. PDS, vol. 12, no. 5, May 2001.

[14] J. Wu, “A Fault-Tolerant and Deadlock-Free Routing Protocol in 2D Meshes Based on Odd-Even Turn Model,” IEEE Trans. Computers, vol. 52, no. 9, Sep. 2003.

[15] 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,” NCA, pp. 52-59, Jul. 2010.

[16] A. Vitkovskiy, V. Soteriou, and C. Nicopoulos, “A highly Robust Distributed Fault-Tolerant Routing Algorithm for NoCs with Localized Rerouting,” INA-OCMC, pp. 29-32, Jan. 2012.

[17] K. Aisopos, A. DeOrio, L. Peh, and V. Bertacco, “ARIADNE: Agnostic Reconfiguration In A Disconnected Network Environment,” PACT, pp. 298-309, 2011.

[18] V. Puente, J. Gregorio, F. Vallejo, and R. Beivide, “Immunet: Dependable Routing for Interconnection Networks with Arbitrary Topology,” IEEE Trans. Computers, vol. 57, no. 12, pp. 1676-1689, Dec. 2008.

[19] A. DeOrio, D. Fick, V. Bertacco, D. Sylvester, D. Blaauw, and J. Hu, “A Reliable Routing Architecture and Algorithm for NoCs” IEEE Trans. CADICS, pp. 726-739, May 2012.

[20] W. Dally, and B. Towles, “Principles and Practices of Interconnection Networks,” Morgan Kaufmann Publisher Inc., San Francisco, CA, 2003.

[21] P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, “Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures,” IEEE Trans. Computers, vol. 54, no. 8, pp. 1025-1040, Aug. 2005

[22] J. Duato, S. Yalamanchili, and L. Ni, “Interconnection Networks: An Engineering Approach,” Morgan Kaufmann Publisher Inc., San Francisco, CA, 2003.

[23] K. Constantinides, et. al., “BulletProof: A Defect-Tolerant CMP Switch Architecture,” HPCA, pp. 5-16, Feb. 2006.

[24] E. Salminen, A. Kulmala, and T. Hamalainen, “Survey of Network-on-chip Proposals,” white paper, OCP-IP, Mar. 2008.

[25] Y. Lu, J. McCanny, and S. Sezer, “Exploring virtual-channel architecture in FPGA based Networks-on-Chip,” in Proceedings of the 24th IEEE International SOC Conference (SOCC), pp. 302-307, Sep. 2011.

[26] Y. Kang, T. Kwon, and J. Draper, “Fault-Tolerant Flow Control in On-Chip Networks,” NOCS, pp. 79-86, May 2010.

[27] A. Banerjee, and S. Moore, “Flow-Aware Allocation for On-Chip Networks,” NOCS, pp. 183-192, May 2009.

[28] W. Liu et al., “A NoC Traffic Suite Based on Real Applications,” VLSI (ISVLSI), IEEE Computer Society Annual Symposium on, Jul. 2011.