Design of Fault-Tolerant and Reliable Networks-on-Chip

Junshi Wang*, Masoumeh Ebrahimi†‡, Letian Huang*, Axel Jantsch§, Guangjun Li*
* University of Electronic Science and Technology of China, Chengdu, China,
Email: wangjsh@std.uestc.edu.cn, {huanglt, gjli}@uestc.edu.cn
† Royal Institute of Technology, Stockholm, Sweden
‡ University of Turku, Turku, Finland, E-mail: masebr@kth.se
§ Vienna University of Technology, Vienna, Austria, E-mail: axel.jantsch@tuwien.ac.at

***Abstract*—Networks-on-Chips (NoCs) are at the core of high performance multi-processor systems-on-chips. As the number of cores and sub-systems on chip grow, the size and complexity of NoCs increase as well. Due to the process variation, aging effects and soft-errors in current and expected future process generations, the probability of failure in the NoCs rises and has to be fought at all levels: circuit, architecture, and communication protocols.**

This paper discusses appropriate fault models for NoCs and their effects on the architecture and network levels. A method to design fault-tolerant NoCs comprising of techniques at the link level, the routing level, and the end-to-end level of the communication is presented. In addition, the proposed method offers an isolation technique where the computing cores are decoupled from the faults in the network. This technique avoids or at least attenuates the severe impacts of faults on the network performance and functionality. These point techniques are combined together to design fault-tolerant and reliable NoCs.

I. Introduction

With technology scaling, process variation, wear-out effects, IR drop, thermal hotspots and other phenomena are increasingly challenging. Rather than striving for a perfect chip, it may be more effective to allow failures sometimes to happen and tolerate them. Three steps are required to provide fault tolerance: detection, diagnosis, correction/reconfiguration. In Network-on-Chip (NoC), fault can be detected and handled at several layers in the communication stack and at different places. Link-level error coding, fault-tolerant routing and end-to-end retransmission are three types of point techniques that are located in different layers and components. Although there is a plethora of publications describing these point techniques [1], attempts to combine several of them into a complete solution are rare.

In this work, we cover detection, diagnosis, and correction and formulate a comprehensive design flow for providing a complete solution for a fault-tolerant NoC (FT NoC). In particular we make the following contributions:

1) A layered fault model is described and a fault notation is proposed to describe the faults at the circuit layer.
2) We propose a comprehensive design flow to cover all types of faults, all communication layers and all components.
3) We demonstrate the flow by realizing a partial FT NoC that covers only the datapath links between routers.
4) We compare fault detection and fault-tolerant methods with respect to their capacity for dealing with transient, intermittent and permanent faults.

II. Related Work

The common goal of all fault-tolerant techniques is to increase the reliability but at the cost of larger area, power, and delay. A fault-tolerant design for NoCs can be discussed at different layers (physical, data link and transport layer) and for different types of faults, i.e. transient, permanent, and intermittent. There are many proposed techniques in literature targeting isolated problems while there are few attempts proposing a comprehensive solution for the reliability of the NoC platform as a whole [1]. This section covers a small portion of different attempts.

Faults may change the content of a packet, for example a bit flip caused by the low noise margin [2], the electromagnetic coupling effects [3], or the crosstalk [4]. The faults on the data path of the routers can be detected and possibly corrected using Error Detecting Codes (EDC) and Error Correcting Codes (ECC) such as hamming codes [5][6]. Complex coding techniques provide a higher correctness and are even capable of correcting burst errors, but the increasing on execution time limits their protection capability [7]. Data redundancy mechanisms such as retransmission at the Link-to-Link or End-to-End levels are also well-known solutions to tackle faulty packets [8][6]. The End-to-End retransmission, however, significantly increases packet latency besides the need of a supporting infrastructure such as an Ack/Nack protocol.

Unlike the faults affecting the content of packets, faults in the control path of the routers are hard to detect and can barely be tolerated. A fault (e.g. stuck-at-1 or stuck-at-0) in the control path may lead to the miscalculation of the routing algorithm or a wrong connection between the input and output ports. When these faults occur, packets may be forwarded to a wrong direction and eventually lead to the deadlock or livelock. A common solution to overcome these types of faults is to disable the faulty components and employ a proper routing algorithm to bypass the disabled faulty links [9][10][11], routers [12], or both[13][14][15]. In almost all of these methods, to deal with the deadlock, the

network operation has to be stopped and the system has to be reset. The new topology is then discovered during the setup period and then the network can start working again under a new configuration. Some proposals avoid the costly reset [15][13][14]. However, some packets might be lost in the transition phase but the network is protected from entering any deadlock situation without being reset.

Redundancy techniques such as the triple modular redundancy or multiple data sampling are also well-known and can be adapted to NoCs [16]. A rigorous analysis on the use of spare wires and cores to replace defective ones is performed in [8], suggesting that the effective yield of the chip and its overall cost can be significantly improved by using redundant modules. The redundancy techniques are popularly used in safety-critical systems and typically avoided in non-critical systems due to their large overhead.

In [16], several approaches such as architectural solutions and retransmission techniques have been proposed and combined together to improve the reliability of the network against transient faults. To avoid costly end-to-end retransmission, the hop-by-hop retransmission scheme is used. In this approach, a flit transmission over a link takes 3 cycles including 1 cycle for detecting an error, 1 cycle for sending an acknowledgement, and 1 cycle for the transmission itself. In addition, to overcome the faults in the router logics, a mechanism to detect the deadlock and recovery from this situation is proposed, which has a considerable latency overhead. In general, although different approaches are carefully designed and combined to satisfy the low area and power overhead, they are not suitable to tackle permanent faults due to large latency penalties.

In Bulletproof [17], different techniques such as triple modular redundancy, end-to-end error detection and resource sparing are combined. Combinations of techniques at different levels such as the system level, component level, and gate level are examined and the tradeoff between area, power, latency, and reliability are reported accordingly. Bulletproof offers a design space exploration but it lacks a comprehensive method satisfying different tradeoffs.

Before dealing with a fault, it must be detected. Many of the fault detection methods rely on off-line testing mechanisms [18][19][20]. However, faults caused by temporary conditions such as overheating, IR-drop, and wear-out can be hardly detected by off-line testing. This is mainly because of the difficulty in creating similar working conditions in a dynamic environment and designing a proper testing mechanism for them. Built-in self-test (BIST) is a common mechanism in digital circuits to detect faults at run-time [9][21][22][19].

NoCAlert [23] provides a comprehensive on-line and real-time fault detection mechanism specific to the NoC platform. It employs a group of lightweight micro-checker modules that collectively implement real-time hardware assertions. The checkers operate concurrently with the normal NoC operation and are capable of detecting a wide range of faults instantaneously.

In this paper, we suggest a combination of fault-tolerant techniques at different layers which are complementary to each other in order to achieve reliability against transient, intermittent and permanent faults at run time. The area, power, and delay overheads are taken into consideration when choosing the techniques.

III. Fault Model

A. Layer structure of fault model

Fault models are categorized according to three different layers (as shown in Table I): physical layer, circuit layer, and network layer.

TABLE I: Layer structure of fault model

Network Layer	Miss-routing, packet loss, deadlock, ...	
Circuit Layer	Behavior	stuck-at-0, stuck-at-1, interval
	Occurrence	transient, intermittent, permanent
	Location	data path, control path
Physical Layer	Thermal, Process variation, IR-drop, Crosstalk, ...	

Physical level. Continuous models describe the physical phenomena leading to misbehavior. They are rooted in the physical laws governing the operation of the devices.

Circuit level. The errors of circuit output value are defined as faults in circuit layer. The faults in circuit layer have three aspects (as shown in Table I). The first aspect, called "value", focuses on the changes of value, including stuck-at-0, stuck-at-1 and reversal. The second aspect, called "Occurrence", describes the time aspect of faults (see examples in Figure. 1). "Location" distinguishes between data path and control path.

SEU (Single Event Upset) and thermal breakdown, lead to functional faults of transistors and result in wrong output values. High temperature, IR-drop, and process variation cause delay fluctuations, which may also result in the errors of output values. Aging phenomena typically start with increasing delays but eventually lead to functional faults.

The relationship between faults in the physical layer and circuit layer are a many-to-many mapping. For example, high temperature can lead to intermittent faults and can also result in permanent damage to a circuit. IR-drop can lead to transient faults when the computation load changes rapidly, or can lead to intermittent faults when the power grid has fluctuations.

Network layer. Network behavior disorder is the abstraction of circuit level faults considering the meaning behind the binary values. Many kinds of network faults have been defined, like mis-routing, packet damage, data error, deadlock, and so on [23].

As network layer faults are highly diverse and depend on the particular behavior and component. Of concern, we have not identified a unified, elegant fault model. Hence, we find network layer fault models are unsuitable to guide the selection and design of NoC fault-tolerant methods. Circuit layer faults are abstractions of physical effects, but there is no simple and direct relation. Since circuit level faults have the virtue to be simple and can be uniformly applied to all components and functions in the network, we use a circuit level fault model as the basis for design and evaluation of a fault-tolerant NoC.

B. Fault models in circuit layer

An error is described as $FM = \{P_O, P_L, P_R, e\}$, in which e is the type of fault, P_O, P_L, and P_R describe the temporal and spatial distribution of faults by defining

the occupation probability, impact probability and recovery probability, respectively. These three parameters can be a constant value or a function. Faults can recover naturally when the causing condition (like high temperature or low voltage) ends. P_R describes duration before the recovery, by defining the expectation of the duration as $\frac{1}{P_R}$. Transient faults usually recover quickly, intermittent faults recovery may take longer, while permanent faults never recover. P_L is only relevant to intermittent faults and describes the probability that the fault actually changes the value of a signal. When an intermittent fault occurs and if P_L is 0.5, there is a 50% chance that a value is changed in a particular cycle. Three examples (Fig. 1) of fault notation are given as follow:

Example 1 (Transient fault) $FM_1 = \{1 \times 10^{-4}, 1, 0.9, Inverted\}$. The duration time of this transient fault is $1/0.9 \approx 1.1$ cycles on average.

Example 2 (Intermittent fault) $FM_2 = \{1 \times 10^{-7}, 0.5, 0.0625, Stack-at-0\}$. The duration of this intermittent fault is $1/0.0625 = 16$ cycles on average. During this time, the value is impacted by the fault in half of the cycles on average.

Example 3 (Permanent faults) $FM_3 = \{1 \times 10^{-10}, 1, 0, Stack-at-1\}$. There is a new stack-at-1 fault every 10^{10} cycles on average and the fault never recovers ($P_R = 0$). These faults are enable in every cycle after occurrence ($P_L = 1$).

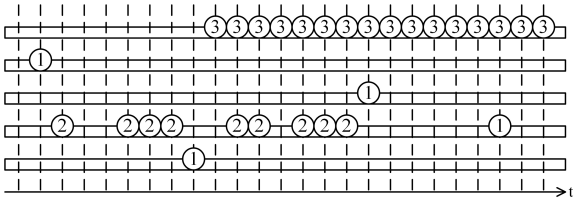

Fig. 1: Fault model examples. A circle means an error. 1-transient faults. 2-intermittent faults. 3-permanent faults

A given signal can suffer from more than one kind of fault. A *fault scenario* $S = \{FM_1, FM_2, FM_3, \ldots\}$ combines the fault notations of different kinds of faults.

IV. Comprehensive Fault-tolerant Design

In this section, we compare several fault detection and fault-tolerant methods and we proposed a comprehensive fault tolerant design flow.

A. Choosing Fault Detection Methods

Fault detection methods are necessary to detect and diagnose faults in the NoC. First, they detect and report faults. Then, they locate the faulty components. For the data path, detection methods should point out the error bits. For the control path, they should be able to locate the faulty component (RC-route compute, VA-virtual buffer allocation, SA-switch allocation, buffers) in a router. Usually, the capacity of location is lower than the capacity of detection.

Three typical fault detection methods are compared in Table II. Error Detection Coding is often used on the data path while assertions can detect errors on the control path by checking certain rules. Build-in Self-Tests (BIST) can provide detailed detection and diagnosis on every component.

TABLE II: Fault Detection and Location Methods
(-: no coverage; P: partial coverage; A: Complete coverage)
(Ref.=Reference, Tran.=Transient, Inter.=Intermittent, Perm.=Permanent)

Method	Ref.	Data Path			Control Path		
		Tran.	Inter.	Perm.	Tran.	Inter.	Per.
Error Detection Code	[5][9]	P	P	P	-	-	-
Assertion	[23]	-	-	-	A	A	A
Build-In Self Test	[9]	P	P	A	P	P	A
	[21]	P	P	A	P	P	A
	[22]	P	P	A	P	P	A
	[19]	P	P	A	-	-	-

When choosing a fault detection method, the impact of the testing procedure is also an important metric. ECC and assertions introduce additional delays, and BISTs have to isolate the components under tests with wrappers. The method proposed by [9][21] isolates one router with its ports while [22] and [19] totally stop the network before the testing. When only testing the link, [9] blocks the packets on the link.

B. Choosing Fault-tolerant Methods

Fault-tolerant methods can be classified into real-time tolerance, reconfiguration and retry.

Real-time tolerance. Methods correct faults at each cycle, e.g. error correction coding (ECC) and multi-sampling (MS). These methods have the capacity for fault detection and fault tolerance at the same time and they can cover transient, intermittent and permanent faults.

Reconfiguration. Faults are tolerated using the redundant resources in the network by reconfiguring routers or paths. Typical methods of this kind include triple modular redundancy (TMR), spare-wire (SW), split-transmission (ST), and fault-tolerant routing algorithms (FTR). To use these methods, faults must already have been diagnosed, hence they cannot work for transient faults.

Retry tolerance. Hop-to-hop retransmission and end-to-end retransmission cannot tolerate any persistent faults. Instead, they wait for the faults to disappear. Similarly, stochastic routing tries different paths at the same time and expects one of them can arrive correctly.

Typical methods are compared in Table III.

TABLE III: Fault-tolerant Methods
(-: no coverage; P: partial coverage; A: Complete coverage)
(Ref.=Reference, Tran.=Transient, Inter.=Intermittent, Perm.=Permanent)

Method	Ref.	Data Path			Control Path		
		Tran.	Inter.	Perm.	Tran.	Inter.	Per.
Error Correct Code	[5][9]	P	P	P	-	-	-
Fault-tolerant Routing	[24]	-	P	A	-	P	A
	[11][15][25]	-	P	P	-	P	P
Triple Modular Redundancy	[26]	P	P	P	P	P	P
Multi Samples	[26]	P	P	P	P	P	P
Stochastic Routing	[27]	A	A	A	A	A	A
Split-Link transaction	[6]	-	P	P	-	-	-
Hop-to-Hop Retransmission	[16]	A	A	-	A	A	-
End-to-End Retransmission	[16]	A	A	-	A	A	-

C. Comprehensive Design

As shown in table II and table III, using only one method cannot address all challenges on reliability. It is necessary to

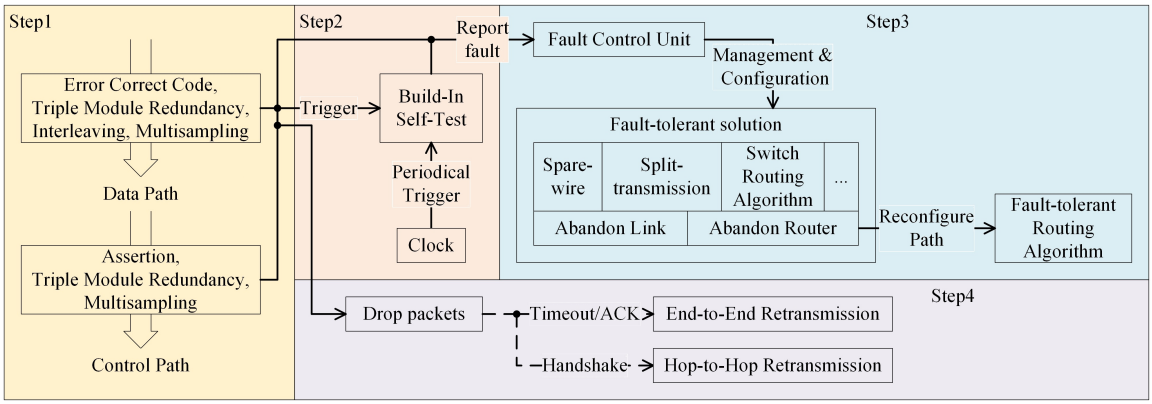

Fig. 2: General flow of fault tolerance

combine different fault detection and fault-tolerant methods following the general flow shown in Figure 2.

The fault-tolerant flow contains four steps. At first, the faults in flits and control signals are detected and, if possible, corrected by real-time tolerance. Error correcting codes (on the data path), assertions (on the control path), triple modular redundancy and multi-sampling can be used in this step.

When faults are detected, a fault diagnosis process (step 2 in Figure 2) is triggered. It can also be triggered periodically. BIST is the most common method to run exhaustive tests.

Faults detected by the first two steps will be reported to the fault control unit. The fault control unit manages and configures different fault-tolerant methods. The fault control unit can be deployed centrally or distributively and located in each router and even each port. Reconfiguration methods will be called in this step. With increasing number of faults in the network, the strategy moves from router reconfiguration to path reconfiguration, because router configuration can maintain the topology but its fault-tolerance capacity is limited. If the faults cannot be recovered by router reconfiguration, fault-tolerant routing algorithms are called to reconfigure paths by abandoning links or routers.

The fourth step deals with faulty flits. Damaged flits have to be dropped and retransmitted. The fault-tolerant flow only drops packets. Retransmission is activated by a time-out of not acknowledged packets or handshake signals.

To design reliable FT NoCs, all four steps are necessary but in each step only one or two candidate methods are chosen. For example, designers can choose one method for the data path and one method for the control path protection in step one. Either hop-to-hop retransmission or end-to-end retransmission is enough to make sure that no packets are lost.

V. Evaluation

We simulated a fault-tolerant design for an 8×8 mesh network where we implemented part of the comprehensive design flow, as shown in Figure 3. The FT NoC tolerates faults on the links between routers. It does not provide a complete solution for a FT NoC but it serves to demonstrate the comprehensive design flow and studies the effectiveness of several methods with respect to transient, intermittent and permanent faults.

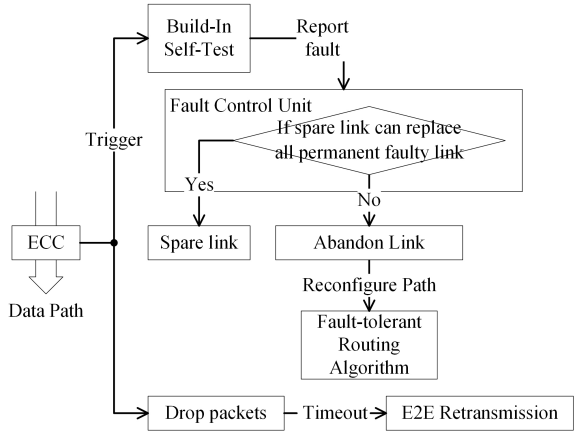

Fig. 3: Fault tolerance flow of simulation

A. Reliable Design

The fault-tolerant flow of our design is shown in Figure 3, which is a simplified version of general flow in Figure 2.

ECC. We implement a group Hamming(12,8) at each port, which can tolerate 1 faulty bit and detect 2 faulty bits in 12 bits [6]. This code word will be checked at each router. If there is only one faulty bit in a group of 12 bits, the codec will correct it. If there are two faulty bits, the codec will report faults to trigger a BIST and drop the packet.

BIST. Test vectors are injected from the output port and checked at the input port. Every port reuses the same testing process.

Fault-tolerant Solution. We select spare-wires and a fault-tolerant routing algorithm. The spare-wire architecture can replace 2 permanently faulty wires in 16 bits wires. A fault-tolerant routing algorithm, called HiPFaR [15], is designed to tolerate faulty links.

Fault Control Unit. The fault control unit is located at each port and makes distributed decisions. If the BIST diagnoses faults, the faulty wires are replaced by spare wires. If spare wires cannot tolerate all permanent faulty wires in a link, this link is marked as faulty and bypassed by the fault-tolerant routing algorithm.

Retransmission. End-to-End retransmission is triggered by

a time-out. To reduce the number of retransmission packets and avoid congestion, only the first packets are retransmitted with timer overflow. In the simulation the time-out is set as 1000 cycles.

The original width of a flit is 128 bits and extended to 192 bits due to the Hamming(12,8) code protection. Including spare wires, the number of wires between two routers is 216 bits. The redundancy rates with and without spare-wire are 33.33% and 40.74% respectively.

B. Simulation Setup

The 8×8 network is simulated using a modified POPNET simulator. Each router contains five physical ports (Local, North, South, East, West). Ports on the Y-axis have two virtual channels and ports on X-axis have one virtual channel. Each virtual channel provides buffers for 12 flits at the input port. The network is simulated with a low injection of 0.01 packets/cycle/router in a random, uniform traffic pattern. A packet contains five flits.

Three fault models are exercised, each with only one kind of faults, transient, intermittent or permanent: $FM_t = \{P_O, 1, 0.9, Inverted\}$, $FM_i = \{P_O, 0.01, 0.001, Inverted\}$ and $FM_p = \{P_O, 1, 0, Inverted\}$, respectively. For the permanent case, we simulate the steady state after the faults have accumulated to compress simulation time. Faults are injected according to P'_O, which is defined as the proportion of accumulated faults.

The following methods are simulated for comparison: 1) ECC; 2) ECC+FTR (Fault-tolerant Routing); 3) ECC+SW (Spare-wire); 4) ECC+SW+FTR, 5) ECC+SW+RT (Retransmission); 6) ECC+SW+RE+FTR. If SW is not used, any line with a permanent fault will be abandoned. XY-routing is implied in those methods without fault-tolerant routing algorithm.

The evaluation metrics are end-to-end latency, delivery rate, and retransmission time. End-to-end latency is the time from when a packet is injected into the network first time until this packet is successfully received, including all retransmissions. To measure the number of successful delivered packets, 30000 packets are injected into the network.

C. Simulation under Transient and Intermittent Fault

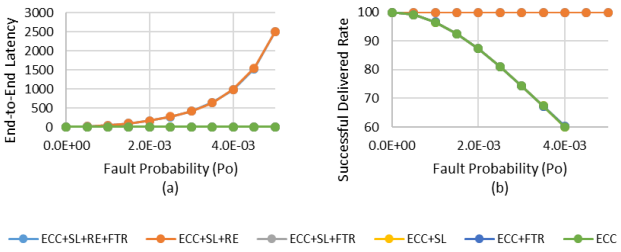

Fig. 4: Simulation with transient fault $FM = \{P_O, 1, 0.9, Inverted\}$.

Figure 4 and Figure 5 illustrate the performance under transient and intermittent faults, respectively. Because spare-wire and fault-tolerant routing are used only when there is any permanent fault, the differences between the lines in each figure are due to retransmission. Thus, in the plots only two distinct lines are visible.

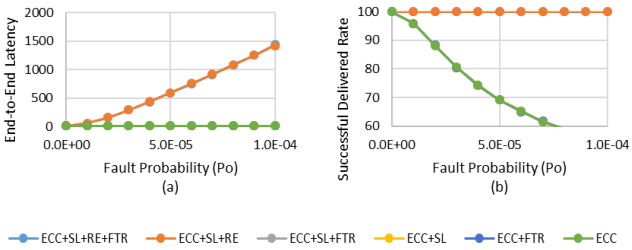

Fig. 5: Simulation with intermittent fault $FM = \{P_O, 0.01, 0.001, Inverted\}$.

Retransmission can provide 100% delivery rate when facing transient faults $P_O < 0.005$ and intermittent faults $P_O < 0.0001$. However, for transient faults with $P_O > 0.004$ and for intermittent faults with $P_O > 7.5 \times 10^{-5}$, the latency becomes excessive.

D. Simulation under Permanent Fault

In contrast to transient faults and intermittent faults, permanent faults accumulate over time. Faults are injected only at the beginning of the simulation according to the given P'_O.

The simulation results are drawn in Figure 6. At first, SW significantly increases the fault-tolerance capacity in the network. The delivery rate of ECC and ECC+FTR drops lower than 60% for $P'_O > 0.008$ and $P'_O > 0.013$, respectively. The methods with SW achieve delivery rates higher than 90% up to $P'_O = 0.028$.

Comparing methods with and without FTR, the number of delivered packets can be increased by reconfiguring paths. In the best observed case, when $P'_O = 0.04$, 13% more packets are delivered due to FTR. Moreover, FTR can reduce the number of retransmissions and the end-to-end latency by choosing another link at the second time.

At the beginning of the simulation, the network knows nothing about faults and discovers them using ECC and BIST. So some packets are rescued by retransmission. On the other hand, the improvement is limited. The original paths and retransmission paths of most packets are the same, because the faults do not change during simulation. With $P'_O = 0.04$, only 0.6% packets are delivered due to retransmission.

Finally, comparing ECC+SW+RE+FTR to ECC+SW+RE with $P'_O > 0.028$, no improvement is observed, because of deadlocks. Thus, to combine FTR with other methods, deadlocks and livelocks must be avoided or handled properly.

VI. Conclusion

Fault-tolerant design takes an important role in NoCs to overcome failures introduced by aggressive technology scaling. In this paper, we have introduced a design flow for FT NoCs and we have reported a first study in combining several methods. From the simulation results, we draw the following conclusions: 1) Retransmission can help to increase the number of delivered packets significantly for transient and

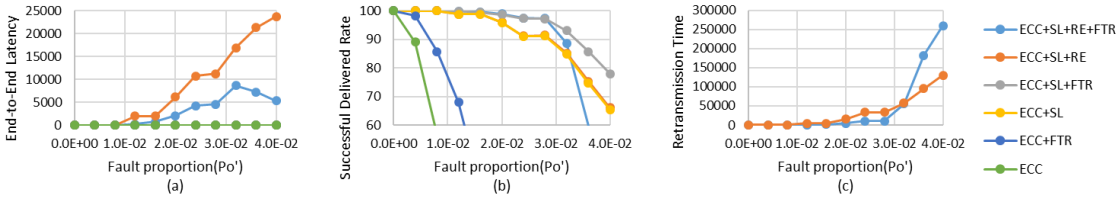

Fig. 6: Simulation with permanent fault

intermittent faults. 2) Spare-wire clearly increases the fault-tolerance capability but has significant overhead. 3) Fault-tolerant routing helps to tolerate more faults and reduces the overhead of retransmission and end-to-end latency, if deadlocks and livelocks are avoided.

ACKNOWLEDGMENT

The research is support by National Natural Science Foundation of China No. 61006027 and No. 61176025, New Century Excellent Talents Program No. NCET-10-0297, the Fundamental Research Funds for the Central Universities No. ZYGX2012J003.

REFERENCES

[1] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for fault tolerance in networks-on-chip," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, p. 8, 2013.

[2] S. Yang and M. Greenstreet, "Noise margin analysis for dynamic logic circuits," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2005, pp. 406–412.

[3] I. Erdin, M. Nakhla, and R. Achar, "Circuit analysis of electromagnetic radiation and field coupling effects for networks with embedded full-wave modules," *IEEE Transactions on Electromagnetic Compatibility*, vol. 42, no. 4, pp. 449–460, 2000.

[4] Aniket and R. Arunachalam, "A novel algorithm for testing crosstalk induced delay faults in vlsi circuits," in *18th International Conference on VLSI Design*, 2005, pp. 479–484.

[5] H. Zimmer and A. Jantsch, "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip," in *First IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2003, pp. 188–193.

[6] Q. Yu and P. Ampadu, "A dual-layer method for transient and permanent error co-management in noc links," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 1, pp. 36–40, 2011.

[7] J. Collet, "A brief overview of the challenges of the multicore roadmap," in *Proceedings of the 21st International Conference on Mixed Design of Integrated Circuits Systems (MIXDES)*, 2014, pp. 22–29.

[8] S. Shamshiri, A.-A. Ghofrani, and K.-T. Cheng, "End-to-end error correction and online diagnosis for on-chip networks," in *IEEE International Test Conference*, 2011, pp. 1–10.

[9] A. DeOrio, D. Fick, V. Bertacco, D. Sylvester, D. Blaauw, J. Hu, and G. Chen, "A reliable routing architecture and algorithm for nocs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 5, pp. 726–739, 2012.

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

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

[12] M. Koibuchi, H. Matsutani, H. Amano, and T. M. Pinkston, "A lightweight fault-tolerant mechanism for network-on-chip," in *Proceedings of the second ACM/IEEE international symposium on networks-on-chip*, 2008, pp. 13–22.

[13] C. Feng, Z. Lu, A. Jantsch, J. Li, and M. Zhang, "A reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning for network-on-chip," in *Proceedings of the Third International Workshop on Network on Chip Architectures*, 2010, pp. 11–16.

[14] M. Ebrahimi and M. Daneshtalab, "A light-weight fault-tolerant routing algorithm tolerating faulty links and routers," *Computing*, pp. 1–18, 2013.

[15] M. Ebrahimi, M. Daneshtalab, and J. Plosila, "High performance fault-tolerant routing algorithm for noc-based many-core systems," in *21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2013, pp. 462–469.

[16] S. Murali, T. Theocharides, N. Vijaykrishnan, M. J. Irwin, L. Benini, and G. D. Micheli, "Analysis of error recovery schemes for networks on chips," *IEEE Design & Test of Computers*, vol. 22, no. 5, pp. 434–442, 2005.

[17] K. Constantinides, S. Plaza, J. Blome, B. Zhang, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky, "Bulletproof: a defect-tolerant cmp switch architecture," in *The 12th International Symposium on High-Performance Computer Architecture*, 2006, pp. 5–16.

[18] J. Raik, V. Govind, and R. Ubar, "An external test approach for network-on-a-chip switches," in *15th Asian Test Symposium*, 2006, pp. 437–442.

[19] E. Cota, F. Kastensmidt, M. Cassel, M. Herve, P. Almeida, P. Meirelles, A. Amory, and M. Lubaszewski, "A high-fault-coverage approach for the test of data, control and handshake interconnects in mesh networks-on-chip," *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1202–1215, 2008.

[20] A. M. Amory, E. Brião, É. Cota, M. Lubaszewski, and F. G. Moraes, "A scalable test strategy for network-on-chip routers," in *IEEE International Test Conference (ITC)*, 2005, pp. 1–9.

[21] M. R. Kakoee, V. Bertacco, and L. Benini, "At-speed distributed functional testing to detect logic and delay faults in nocs," *IEEE Transactions on Computers*, vol. 63, no. 3, pp. 703–717, 2014.

[22] X.-T. Tran, Y. Thonnart, J. Durupt, V. Beroulle, and C. Robach, "Design-for-test approach of an asynchronous network-on-chip architecture and its associated test pattern generation and application," *IET Computers & Digital Techniques*, vol. 3, no. 5, pp. 487–500, 2009.

[23] A. Prodromou, A. Panteli, C. Nicopoulos, and Y. Sazeides, "Nocalert: An on-line and real-time fault detection mechanism for network-on-chip architectures," in *45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2012, pp. 60–71.

[24] E. Wachter, A. Erichsen, A. Amory, and F. Moraes, "Topology-agnostic fault-tolerant noc routing method," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2013, pp. 1595–1600.

[25] H. Hsin, E. Chang, C. Lin, and A.-Y. Wu, "Ant colony optimization-based fault-aware routing in mesh-based network-on-chip systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 11, pp. 1693–1705, 2014.

[26] P. A. Frantz, M. Cassel, F. L. Kastensmidt, É. Cota, and L. Carro, "Crosstalk-and seu-aware networks on chips," *IEEE Design & Test*, vol. 24, no. 4, pp. 340–350, 2007.

[27] P. Bogdan, T. Dumitraş, and R. Marculescu, "Stochastic communication: A new paradigm for fault-tolerant networks-on-chip," *VLSI design*, vol. 2007, 2007.