Adaptive Error- and Traffic-aware Router Architecture for 3D Network-on-Chip Systems

Akram Ben Ahmed, Michael Meyer, Yuichi Okuyama, Abderazek Ben Abdallah
The University of Aizu
Graduate School of Computer Science and Engineering
Aizu-Wakamatsu 965-8580, Japan
E-mail: {d8141104, d8161104, okuyama, benab}@u-aizu.ac.jp

***Abstract*—The advent of deep sub-micron and 3D integration technologies has exacerbated reliability issues in packet-switched on-chip interconnection networks. A lot of researches have been conducted in order to make these systems immune to any short-term malfunction or permanent physical damage while minimizing the performance degradation as much as possible. In this paper, we present an adaptive Error-, and Traffic-aware 3D-NoC router architecture, called 3D-Fault-Tolerant-OASIS (3D-FTO)[1]. 3D-FTO manages to avoid the system failure at the presence of a large number of faults and addresses the fault occurrence in links, input-buffers, and the crossbar, where the faults are more often to happen. The proposed 3D-FTO system was synthesized using Synopsis Design Compiler at 45nm CMOS process technology. Evaluation results show that our 3D-FTO is able to work around different kinds of faults ensuring graceful performance degradation while minimizing the additional hardware complexity and remaining power-efficient.**

***Keywords*-3D NoC; Error-aware; Traffic-aware; Adaptive.**

I. Introduction

During the past few decades, a lot of research has been focusing on Three-dimensional Networks-on-Chips (3D-NoCs) [3], [4], [5], [6], [7], [8] as an auspicious solution to alleviate the interconnect bottleneck and reduce the power consumption in current System-on-Chips (SoCs) designs. As 3D-NoC architectures started to show their performance benefits and energy efficiency against 2D-NoC systems, questions of their reliability to sustain their performance growth began to arise [9]. This is mainly due to challenges inherited from both Three-dimensional Integrated-Circuits (3D-ICs) and NoCs; on one side, the complex nature of 3D-IC fabrics and the continuing shrinkage of semiconductor components. Furthermore, the significant heterogeneity in 3D chips that are more likely to mix logic layers with memory layers adding more complexity and increasing the fault probability in a system [10]. The other challenge is that the single-point-failure nature of NoC introduces a big concern to their reliability as they are the sole communication medium.

[1]This project is partially sponsored by Competitive research funding, Ref. P1-12, Fukushima-Japan, and supported by VLSI Design and Education Center(VDEC), the University of Tokyo Japan, in collaboration with Synopsys Inc.

As a result, 3D-NoC systems are becoming susceptible to a variety of faults caused by crosstalk [11], impact of radiations [12], oxide breakdown [13], and so on [14]. A simple failure in a single transistor caused by one of these factors may compromise the entire system reliability where the failure can be illustrated in corrupted message delivery, time requirements unsatisfactory, or even sometimes the entire system collapse.

Many works have been conducted to tackle the fault-tolerance in NoC systems (2D and 3D) where they can be classified depending on the target system, the fault's type, or the faults' handling mechanism (e.g., using routing algorithms or architectural solutions). We previously presented in [1], [2] some of the well-known routing algorithms used in 3D-NoC systems that focused mainly on link-failure. Another interesting work presented by *Radetzki et al.* [14] gives a survey of the different failure mechanisms, fault models, detection techniques, and recovery methods used in the different layers of a given NoC. In this paper, we focus on works that presented reliable router architectures for 2D-NoC architecture, but can be adopted in the third dimension. For instance, *Constantinides et al.* [15] proposed the BulletProof router which is based on N-modular redundancy (NMR) technique. The NMR method requires the presence of N copies of a given targeted component. Thus, N times extra silicon area is needed; therefore, such method is very expensive in terms of area. Moreover, as the area increases the fault occurrence probability increases, as well. As a result, duplicating components may not lead to endorse the reliability.

Kim et al. [16] proposed RoCo that employs decoupled parallel arbiters and uses smaller crossbars for row and column connections in order to allow the router to be decomposed. Look-ahead routing is used to tolerate faults in the Routing Computation stage (RC). By sharing arbiters from Virtual Channel Allocation stage (VCA), fault tolerance in the Switch Allocation stage (SA) can be ensured. However, this router cannot tolerate faults in VCA and Crossbar Traversal (CT) stages where the area is more important and faults are more likely to occur in these stages.

Poluri et al. [17] presented an improved router design

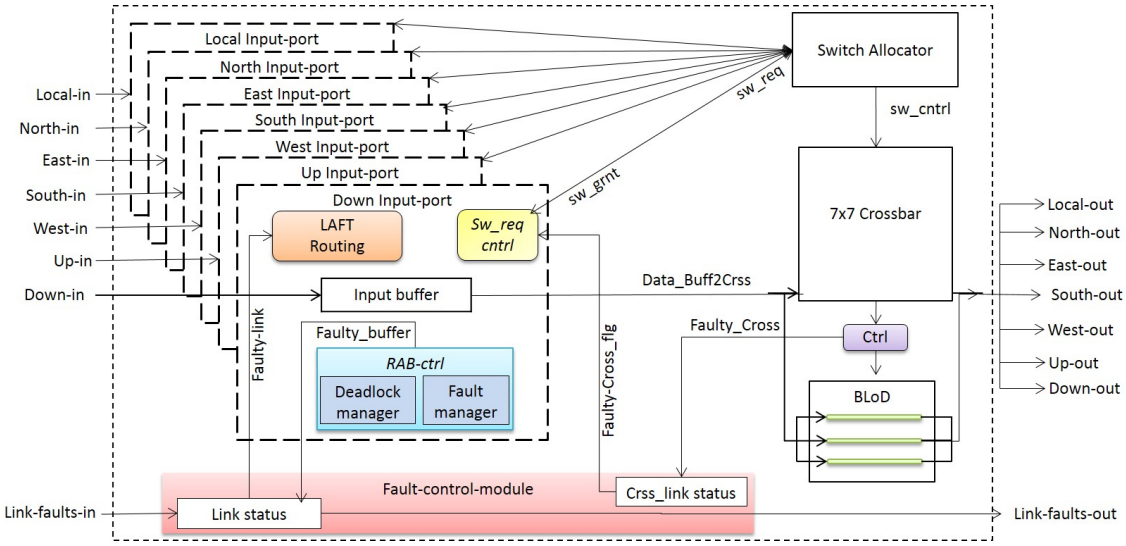

Figure 1. 3D-Fault-Tolerant-OASIS-NoC router architecture.

where they added a small minimal correction circuitry to provide a better fault-tolerance in each one of the pipeline stages. The proposed router adds redundant components and resources sharing to ensure the fault-tolerance in the RC, VCA, SA, and CT pipeline stages. Evaluation results show the outperformance of their proposed router and its higher reliability when compared to earlier works. However, they do not consider the presence of faults in the input buffer. Buffers consume the largest portion of area and power; therefore the probability of fault occurrence is the highest when compared to the other components of the router.

Vicis is another work proposed in [18] targeting permanent faults. The authors presented *flexible-fifo* to deal with permanent faults in the input-buffers. To deal with faults in the crossbar, they introduced *Crossbar-Bypass-Bus* that provides an alternative path when a faulty crossbar link is detected. *Vicis* suffers from three main drawbacks: first, this architecture deals with only permanent faults and does not consider transient and intermittent faults. However, transient faults cause the majority of failures, (80%) [19], while the remaining failures originate mainly in permanent and intermittent faults. On the other hand, this should not diminish the importance of permanent faults; thus, analyzing the three types of failures is imperative to represent the real behavior of 3D-NoC systems. The second drawback is that when multiple faulty crossbar paths are detected, flits from different input-ports should compete the access for this single bypass-bus. This puts under question the scalability of this approach when the latency may increase at the presence of more than one faulty crossbar link. Third, the companion routing protocol contains some restrictions and turns to ensure the system deadlock-freedom; however, as we previously said, the addition of such rules may lead to nonminimal path, thus, increasing the latency of the flit.

II. Router Architecture

Figure 1 represents the high-level representation of 3D-Fault-Tolerant-OASIS (3D-FTO). The baseline router's component are depicted in white and the added enhancements for fault-tolerance and robustness are colored. 3D-FTO router relies on simple recovery techniques based on system adaptivity with redundant structural resources to contain faults' occurrence (in input-buffers, crossbar, and links) and prevent from the system failure, or information corruption or loss.

As shown in Fig.1, 3D-FTO router contains seven input-ports, a switch-allocator, a crossbar, and a Fault-Control-Module (FCM). In this section, we explain the enhancements added in 3D-FTO router including: first, the Random-Access-Buffer (RAB) for deadlock-recovery and fault-tolerance in the input-buffer; second, the Bypass-Link-on-Demand (BLoD) approach to handle multiple faulty channels in the crossbar; finally, the FCM module responsible for the assignment and control of the different detection and recovery tasks to the previously mentioned techniques. Look-Ahead-Fault-Tolerant (LAFT) routing algorithm to tackle link failure was previously presented in [2].

A. Random-Access-Buffer

Figure 2 represents the block diagram of the proposed Random-Access-Buffer mechanism (RAB). RAB was previously presented as an efficient and low-overhead solution to ensure deadlock-freedom [1], [20]. RAB was extended to be able to detect transient, intermittent, and permanent faults in the input-buffer. For the detection, we assume the presence of a module (*fault-detect* in Fig. 2) that checks the buffer entries' fault status. When a fault is detected in one

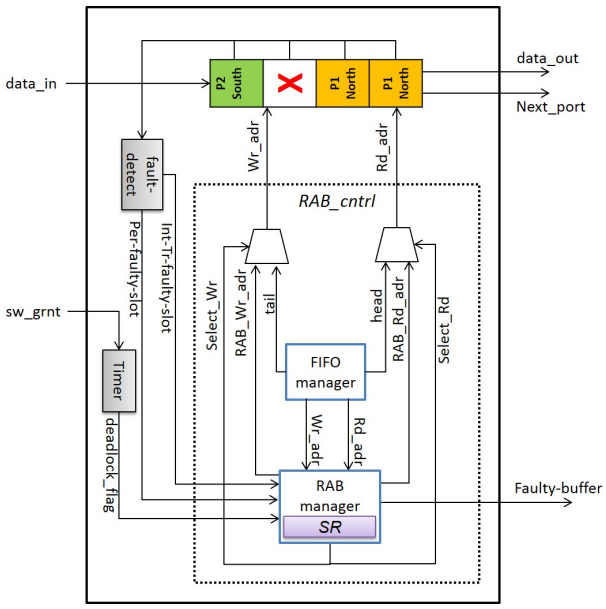

Figure 2. Random-Access-Buffer block diagram.

of the slots, it can send one of two signals: *Int-Tr-faulty-slot* signal to inform the RAB-manager module the presence of transient or intermittent fault. Then, RAB-manager will take into consideration the flagged slots when assigning *Wr-adr* and *Rd-adr* addresses, while keeping checking the flagged slots whether their faults were recovered or not. In the case where a permanent fault is detected, the *fault-detect* sends *Perm-faulty-slot* signal to the RAB-manager module. This information is important because if the RAB-manager finds that only one buffer slot is nonfaulty, and the remaining ones are permanently faulty, it sends *Faulty-buffer* signal to the Fault-control-module to disable the entire input-port and update the *Link-status* array (See Fig. 1).

To keep record of the faulty slots, the *status register* (SR), previously presented for deadlock-recovery, was extended to an array that hosts n 2-bit items (where n is the buffer depth). The value of each item can be *00* to inform that the corresponding flit is not causing the deadlock and the buffer slot is not faulty. *01* indicates that the buffer slot is not faulty but the request of the hosted flit is causing deadlock; therefore, this slot can be consulted again to check whether the deadlock is removed or not, but it cannot be used to store incoming flits to avoid flit overwriting. An element in the status array is updated to *10*, if a transient or intermittent fault is detected. In this case, the slot cannot be consulted nor used to store incoming flits (to avoid additional latency for consulting broken slots). Finally, *11* is used to declare that the corresponding buffer entry is permanently faulty. As we previously said, this information will be used to issue the *Faulty-buffer* signal. In addition to the timer, the RAB mechanism is triggered by the *fault-detect* module where in case of a transient or intermittent fault is removed from a given slot, the *fault-detect* informs the RAB-manager to update the status array of the corresponding slot to *00* so it can be used by other incoming flits.

Figure 3 shows an example how the RAB mechanism works. In each input-port, a RAB-controller (RAB-cntrl) manages the detection of deadlock and faults as well as handling the proper assignment of *Wr-adr* and *Rd-adr* addresses. In Fig. 3 (a), and after a permanent fault was detected (red cross) in one of the buffer slots, the RAB-manager updates the corresponding element in the status array to *11*. Furthermore, by reading the *sw-grnt* signal received from the Switch-allocator, the timer issues a deadlock-flag and the RAB-manager updates the corresponding slot to *01*. The RAB-cntrl reads the head of the next packet in the buffer and checks whether the requested out-port is different from the one previously flagged as blocked or not. When it finds a request whose channel is free, it sends a request to the Switch-allocator to be served. When the request is granted, the flits of the granted packet are read from the buffer and the freed slots can be used to host another incoming packet which is stored in a slot whose value in the status array is *00*. After new flits are written in the buffer, the blocked packet is checked again (Figure 3 (b)). When the RAB-cntrl receives a grant for the direction requested (North), the packet is read from the buffer and the status array is updated to *00*. At the same time, the *fault-detect* module has found an intermittent fault (green cross), then again the status array to *10* to prevent from reading or writing into the corresponding slot before the fault is removed (Figure 3 (c)).

B. Bypass-Link-on-Demand

The Bypass-Link-on-Demand mechanism, depicted in Fig. 4 (a), provides additional escape channels whenever the number of faults in the baseline 7x7 crossbar increases. In this figure, we considered two Bypass-links for simplicity. The *Ctrl* unit, shown in this figure, manages to check the crossbar link status. In the case where a fault is detected in one or several links, it sends flags to the FCM which disables the faulty crossbar links and enables the appropriate number of bypass channels. The easiest approach is to provide a dedicated Bypass-Link for each crossbar channel. In this fashion, both fault-tolerance and performance are guaranteed because the input-ports requests do not share the Bypass-Links, even when all the baseline 7x7 crossbar links are faulty. However, this technique is the same as duplicating the entire crossbar; therefore, additional area and power overhead is certain to occur. Additionally, when the fault-rate is low, only one or two Bypass-Links are enough to handle the requests of the faulty crossbar-links. According to these facts, we decided to perform an incremental approach, where we analyze the used benchmark and the assumed fault-rate and we increment the number of Bypass-Links until the

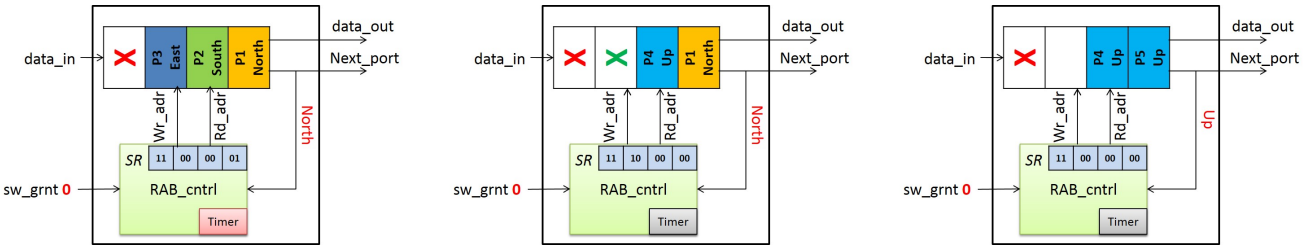

Figure 3. Example of Random-Access-Buffer mechanism for deadlock-recovery and fault-tolerance. Red crosses represent permanent faults, and the green one represents an intermittent or transient fault

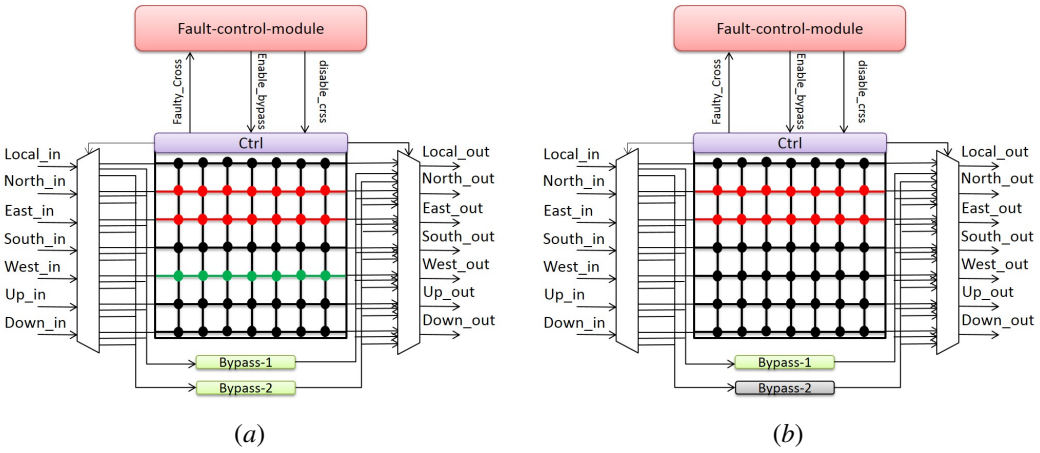

Figure 4. Example of Bypass-Link-on-Demand.

performance is steady or almost unchanged.

The number of Bypass-links is very important and should be minimized as much as possible to reduce the area and power overhead. Therefore, we decided to exploit the unused crossbar-links already existing in the system. These links are the ones located at the edges of the network where there is no neighboring node and, therefore, the corresponding crossbar link is unused. With this optimization, an important area and power saving can be achieved while keeping the performance at its peak.

Assuming the example in Fig. 4 (a) where three faults are detected: two are permanent in the North- and East-links (red), and the other one is transient in the West-link (green). The two available bypass channels are activated to handle the incoming requests from the different input-ports allocated to the faulty crossbar links. After a period of time, the transient fault is removed and the *Ctrl* module sends information to the FCM which re-enables again the West-link and deactivates one of the Bypass-Links (Bypass-2 in Fig. 4 (b) is deactivated (gray)) as it is no longer necessary.

C. Fault-Control

The Fault-control-module (FCM) is one of the main components of our system. This is because it manages the diagnosis and recovery from all kinds of faults in three main components: inter-router links, input-buffers, and the crossbar. Starting with the inter-router links, the link status of each router and those of its neighboring nodes are stored in a small array named *Link-status*. This information is always sent to the LAFT-routing module to be used during the selection of the *Next-port* for the next node.

To handle the faults in the crossbar FCM interacts with the *Ctrl* unit in the crossbar circuit to exchange fault information and control signals. As shown in Fig. 4 (a), the *Ctrl* unit is the medium between the baseline 7x7 crossbar and the additional Bypass-links. The main task of this unit is to detect the presence of faults in the crossbar and to keep informing the FCM about its fault status. The FCM monitors this incoming fault information and stores them in a register named *Crss-link status* (Fig. 1). When a fault is detected, the FCM sends three signals concurrently: two for the *Ctrl* unit to enable one of the Bypass-links and disable the faulty crossbar link, and the third one to the Sw-req-cntrl (Fig. 1) to prevent flits from requesting the faulty crossbar-link and ask the permission to use one of the Bypass-links instead. When the number of faults increases, the FCM manages the fair distribution of the different requests on the available Bypass-links in a fair way. On the other side, the *Ctrl* unit

has the task to enable and disable the Bypass-links for power saving depending on the signals received from the FCM. This means that when the crossbar is valid, the Bypass-links are put asleep to save dynamic power (as represented in Fig. 4 (b)). When faults are detected, the *Ctrl* unit awakens the appropriate number of Bypass-links depending on the information received from the FCM.

III. Evaluation

A. Evaluation methodology

Our proposed 3D-Fault-Tolerant-OASIS (3D-FTO) system was designed in Verilog-HDL, and synthesized using Synopsis Design Compiler with 45nm technology library [21]. We evaluate the hardware complexity of LAFT router in terms of area utilization and power consumption (static and dynamic). To evaluate the performance of the proposed system, we selected Matrix-multiplication [22] and JPEG-encoder [23] as real benchmarks and also two traffic patterns: Transpose [24] and Uniform [25].

Using these four benchmarks, we evaluated the latency/flit and throughput of the proposed 3D-FTO system under each of the aforementioned applications. The obtained results are compared with the baseline [6], [7] and XYZ-based [26] routers. We observed the performance variation of 3D-FTO under different fault-rates of link, crossbar-link, and buffer-slots (0%, 5%, 10%, and 20%). We set the number of Bypass-links in BLoD to three as it seemed to be the best tradeoff between performance and complexity. For the input-buffer, we set the buffer depth to four and employed the RAB mechanism. During the evaluation, we divided the faults into three portions: the biggest portion is allocated for transient faults and the remaining two smaller portions are considered for permanent faults and intermittent faults according to the assumption made in [19]. We set the fault-rate for each one of the targeted components (input-buffer, crossbar, and links) proportionally to their corresponding percentage of the entire router's area. The number of links, crossbar-links, and buffer slots can be calculated using formula (1) [4], (2), and (3), respectively:

$$\#links = N_1 \times N_2 \times (N_3 - 1) + N_1 \times N_3 \times (N_2 - 1) + N_2 \times N_3 \times (N_1 - 1) \quad (1)$$

$$\#Crossbar\_links = OP \times N_1 \times N_2 \times N_3 \quad (2)$$

$$\#Buffer\_slots = BD \times IP \times N_1 \times N_2 \times N_3 \quad (3)$$

Where N_1, N_2 and N_3 are the respective network's X, Y and Z dimensions. OP is the number of output-ports, IP is the number of input-ports, and BD is the buffer depth.

Table I
Simulation configuration.

Parameters / System		XYZ-based	Baseline	3D-FTO
Network Size (Mesh)	JPEG	3×3×3	3×3×3	3×3×3
	Matrix	3×6×6	3×6×6	3×6×6
	Transpose & Uniform	4×4×4	4×4×4	4×4×4
Flit size	JPEG	27 bits	30 bits	30 bits
	Matrix	31 bits	34 bits	34 bits
	Transpose & Uniform	31 flit	34 flit	34 flit
Header size	JPEG	10 bits	13 bits	13 bits
	Matrix	10 bits	13 bits	13 bits
	Transpose & Uniform	10 bits	13 bits	13 bits
Payload size	JPEG	16 bits	16 bits	16 bits
	Matrix	21 bits	21 bits	21 bits
	Transpose & Uniform	21 bits	21 bits	21 bits
Buffer Depth		4	4	4
Routing		XYZ	LA-XYZ	LAFT

B. Evaluation results

1) Latency per flit evaluation: The results of the latency/flit evaluation are depicted in Fig. 5. We can see that in the absence of faults, and thanks to the employed LAFT routing, 3D-FTO system has the best performance when compared to XYZ- and the LA-XYZ- based systems, even at 5% (Uniform and Matrix) or 10% (Transpose) fault rates. This latency/flit reduction can reach an average of 37% and 18.5% when compared to the XYZ- and LA-XYZ-based systems, respectively. When the fault-rate increases in the three components (link, crossbar, and input-buffer) 3D-FTO's latency increases, as well. However, in some applications (Transpose and Matrix) 3D-FTO still performs better than XYZ-based system (no fault consideration) even at 20% fault-rate, but higher latency than that of the baseline LA-XYZ-based design. With the Uniform and JPEG applications, the latency degradation is more important. it can reach an average of 12.1% and 31.7% when compared to XYZ- and the LA-XYZ based systems. This performance degradation is caused mainly by the nonminimal routing required in such communication types. In JPEG and Uniform, neighboring nodes tend to communicate between each other. A single fault in a buffer-slot or a crossbar-link will not considerably affect the system performance; however, a single faulty-link causes nonminimal routings. As a consequence, additional clock cycles are necessary to perform the rerouting. But, with other applications (Transpose or Matrix) exhibiting long distance communications, 3D-FTO performs better or almost the same as XYZ-based system when considering a 20% fault-rate.

2) Throughput evaluation: The throughput evaluation results are shown in Fig. 6. The 3D-FTO system exhibits the best throughput when compared to the other two systems in the absence of faults. This throughput outperformance can reach the 51% and 38% when compared to XYZ- and LA-XYZ- based systems, respectively. Even in the presence of faults, 3D-FTO still maintains a sustainable throughput, and as we increase the fault-rate the throughput degrades gracefully. This is justified by the fact that 3D-FTO

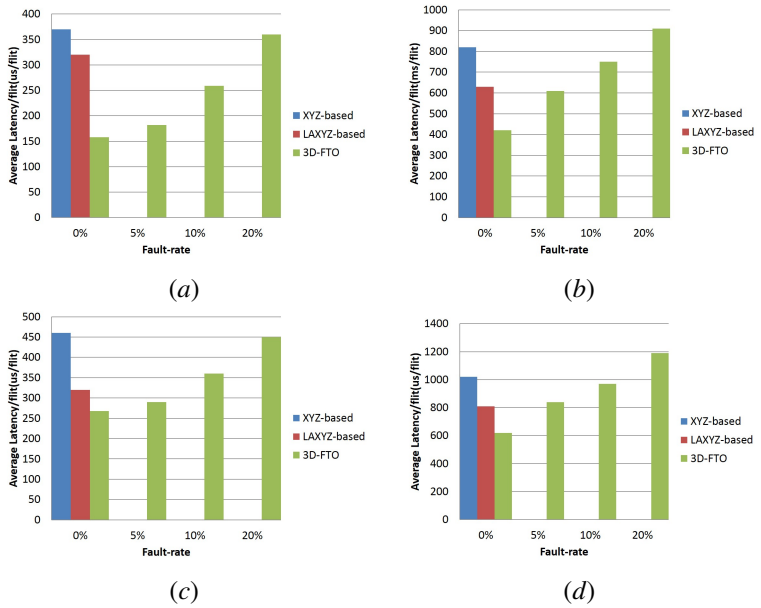

Figure 5. 3D-Fault-Tolerant-OASIS latency/flit evaluation with: (a) Transpose; (b) Uniform; (c) 6 × 6 Matrix; (d) JPEG.

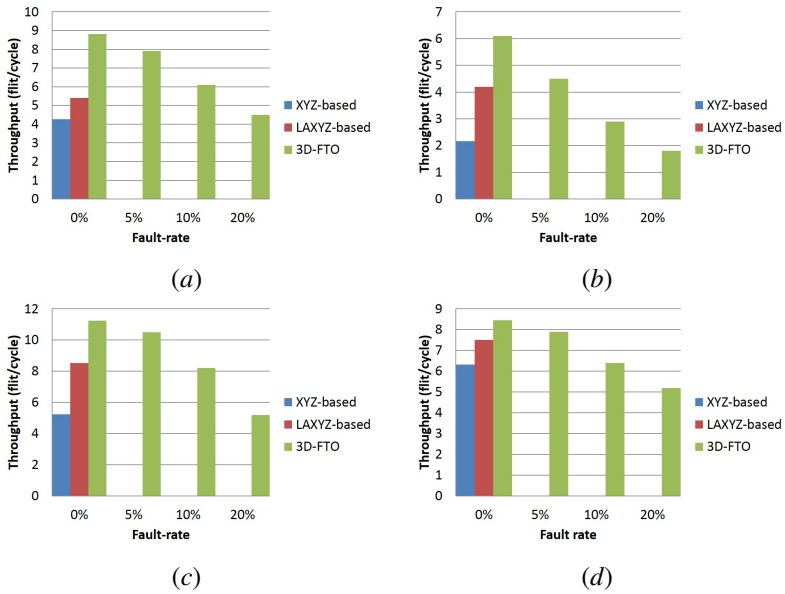

Figure 6. 3D-Fault-Tolerant-OASIS throughput evaluation with: (a) Transpose; (b) Uniform; (c) 6 × 6 Matrix; (d) JPEG.

provides higher throughput than XYZ-based system even when considering 20% fault-rate (Transpose and Matrix-multiplication). When running Uniform and JPEG-encoder, 3D-FTO provides an average of 11.2% and 30% less throughput than that of XYZ- and LA-XYZ- based systems, respectively. It is important to mention that neither XYZ- or LA-XYZ- based systems support fault-tolerance, and any single failure may lead to corrupted information or the entire system crash.

3) Complexity Evaluation: In our final evaluation, we considered the hardware complexity of the proposed 3D-FTO. Table II illustrates the hardware complexity results of 3D-FTO in terms of area and power (static+dynamic) when compared to the baseline router [6], [7]. From this table, we can see that the proposed router requires 38.3% additional area and 32.6% increased power. It is important to mention that the power overhead became less important when connecting all the modules together and disabling the unused components.

Table II
ROUTER HARDWARE COMPLEXITY EVALUATION RESULTS.

System / Parameter	Area μm	Total Power μW
Baseline	7654	886.32
3D-FTO	10587	1175.6

IV. CONCLUSION AND FUTURE WORK

In this paper, we present a Fault- and Traffic-aware 3D-NoC router architecture, called 3D-Fault-Tolerant-OASIS (3D-FTO). 3D-FTO manages to avoid the system failure in the presence of a large number of faults, while ensuring graceful performance degradation and minimizing the additional hardware complexity and remaining power-efficient. In addition to Look-Ahead-Fault-Tolerant routing algorithm, previously presented to tackle the faulty-links problem, the proposed architecture is leveraging on reconfigurable components to handle the fault occurrence in the input-buffers thanks to a smart mechanism, called Random-Access-Buffer (RAB). Moreover, a technique named Bypass-Link-on-Demand was introduced to relieve the congestion caused by faults in the crossbar.

From the performance evaluation, the proposed system still performs better than XYZ-based system with Transpose and Matrix-multiplication applications, even at 20% fault-rate. In terms of hardware complexity, 3D-FTO exhibits 38.3% additional area and 32.6% power overhead when compared to the baseline LA-XYZ-based system. The power overhead could be controlled thanks to the power-management employed in 3D-FTO that is based on disabling the unused components and faulty input-ports.

As a future work, we want to study the possibility of implementing a fault-detection mechanism capable of detecting different kinds of faults at runtime with no considerable area or latency overhead. In addition, we plan to perform an in-depth study of thermal power to observe 3D-FTO's behavior with such important parameter.

REFERENCES

[1] A. Ben Ahmed and A. Ben Abdallah, "Graceful Deadlock-Free Fault-Tolerant Routing Algorithm for 3D Network-on-Chip Architectures", Journal of Parallel and Distributed Computing, Vol. 74-4, pp. 2229-2240, Apr. 2014.

[2] A. Ben Ahmed and A. Ben Abdallah, "Architecture and Design of High-throughput, Low-latency, and Fault-Tolerant Routing Algorithm for 3D-Network-on-Chip (3D-NoC)", The Journal of Supercomputing, Vol. 66-3, pp. 1507-1532, Dec. 2013.

[3] X. Wu *et al.*, "Electrical Characterization for Intertier Connections and Timing Analysis for 3-D ICs", IEEE Trans. on Very Large Scale Integration (VLSI) Systems, Vol. 20-1, pp. 186-191, Jan. 2012.

[4] B. Feero and P. P. Pande, "Performance Evaluation for Three-Dimensional Networks-on-Chip", Proc. of IEEE Computer Society Annual Symp. on VLSI (ISVLSI), pp. 305-310, May 2007.

[5] A. Ben Ahmed, A. Ben Abdallah and K. Kuroda, "Architecture and Design of Efficient 3D Network-on-Chip (3D NoC) for Custom Multicore SoC", IEEE Proc. of the 5th Int. Conf. on Broadband, Wireless Computing, Communication and Applications, pp. 67-73, November 2010.

[6] A. Ben Ahmed and A. Ben Abdallah, "LA-XYZ: Low Latency, High Throughput Look-Ahead Routing Algorithm for 3D Network-on-Chip (3D-NoC) Architecture", The 6th IEEE Int. Symp. on Embedded Multicore SoCs, pp. 167-174, Sept. 2012.

[7] A. Ben Ahmed and A. Ben Abdallah, "Low-overhead Routing Algorithm for 3D Network-on-Chip", IEEE Proc. of The Third Int. Conf. on Networking and Computing pp. 23-32, December 2012.

[8] A. Ben Abdallah, "Multicore Systems-on-Chip: Practical Hardware/Software Design", 2nd Edition, Publisher: Atlantis Press, 2013, ISBN-13:978-9491216916.

[9] L. Benini and G. De Micheli, "Networks on Chips: Technology and Tools", Morgan Kauffmann, 2006.

[10] I. Loi *et al.*, "Characterization and Implementation of Fault-Tolerant Vertical Links for 3-D Networks-on- Chip", IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 30-1, pp. 124-134, Jan. 2011.

[11] M. Cuviello, S. Dey, X. Bai, and Y. Zhao, "Fault modeling and simulation for crosstalk in system-on-chip interconnects", In IEEE/ACM Int. Digest of Technical Papers on Computer-Aided Design, pp. 297-303, 1999.

[12] S. Borkar, "Designing reliable systems from unreliable components: The challenges of transistor variability and degradation" IEEE Micro, Vol. 25-6, pp. 10-16, Nov.-Dec. 2005.

[13] T. Kuroi *et al.*, "Sub-Quarter-Micron Dual Gate CMOSFETs with Ultra-Thin Gate Oixde of 2nm", Symp. on VLSI Technology, pp.210-211, Jun. 1996.

[14] M. Radetzki *et al.*, "Methods for Fault Tolerance in Networks-on-Chip", ACM Computing Surveys (CSUR), Vol. 46-1, pp. 1-38, Oct. 2013.

[15] K. Constantinides *et al.*, "BulletProof: A defect-tolerant CMP switch architecture", Proc. of the 12th Int. Symp. on High-Performance Computer Architecture (HPCA), pp. 5-16, Feb. 2006.

[16] J. Kim *et al.*, "A Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks", Proc. of the 33rd Int. Symp. on Computer Architecture (ISCA), pp. 4-15, Jun. 2006.

[17] P. Poluri and A. Louri, "An Improved Router Design for Reliable On-Chip Networks", Proc. of the 25th Int. Symp. on Computer Architecture and High Performance Computing (SBAC-PAD), pp. 49-56, Oct. 2013.

[18] A. DeOrio *et al.*, "A Reliable Routing Architecture and Algorithm for NoCs", IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 31-5, pp. 726-739, May 2012.

[19] T. Lehtonen, P. Liljeberg and J. Plosila, "Online Reconfigurable Self-timed links for Fault Tolerant NoC", VLSI Design, Vol. 2007, pp. 1-13, 2007.

[20] A. Ben Ahmed and A. Ben Abdallah, "Fault-tolerant Routing Algorithm with Deadlock Recovery Support for 3D-NoC Architectures", The 7th IEEE International Symposium on Embedded Multicore SoCs, pp. 67-72, September 2013.

[21] Nangate 45nm open cell library, http://www.nangate.com.

[22] P. Chan *et al.*, "The Parallel Algorithm Implementation of Matrix Multiplication Based on ESCA", IEEE Asia Pacific Conf. on Circuits and Systems, pp. 1091-1094, Dec. 2010.

[23] Y. L. Lee, J. W. Yang, and J. M. Jou, "Design of a distributed JPEG encoder on a scalable NoC platform", IEEE Int. Symp. VLSI-DAT, pp. 132-135, Apr. 2008.

[24] A. A. Chien and J. H. Kim, "Planar-Adaptive Routing: Low-Cost Adaptive Networks for Multiprocessors", Journal of the ACM, Vol. 42-1, pp. 91-123, Jan 1995.

[25] R. Sivaram, "Queuing delays for uniform and nonuniform traffic patterns in a MIN", ACM SIGSIM Simulation Digest, Vol. 22-1, pp. 17-27, 1990.

[26] H. Sullivan and T. R. Bashkow, "Large Scale, Homogeneous, Fully Distributed Parallel Machine", Annual Symposium on Computer Architecture, ACM Press, pp. 105-117, Mar. 1977.