A low-overhead soft–hard fault-tolerant architecture, design and management scheme for reliable high-performance many-core 3D-NoC systems

Khanh N. Dang · Michael Meyer · Yuichi Okuyama · Abderazek Ben Abdallah

Published online: 13 January 2017
© Springer Science+Business Media New York 2017

Abstract The Network-on-Chip (NoC) paradigm has been proposed as a favorable solution to handle the strict communication requirements between the increasingly large number of cores on a single chip. However, NoC systems are exposed to the aggressive scaling down of transistors, low operating voltages, and high integration and power densities, making them vulnerable to permanent (hard) faults and transient (soft) errors. A hard fault in a NoC can lead to external blocking, causing congestion across the whole network. A soft error is more challenging because of its silent data corruption, which leads to a large area of erroneous data due to error propagation, packet re-transmission, and deadlock. In this paper, we present the architecture and design of a comprehensive soft error and hard fault-tolerant 3D-NoC system, named 3D-Hard-Fault-Soft-Error-Tolerant-OASIS-NoC (3D-FETO). With the aid of efficient mechanisms and algorithms, 3D-FETO is capable of detecting and recovering from soft errors which occur in the routing pipeline stages and leverages reconfigurable components to handle permanent faults in links, input buffers, and crossbars. In-depth evaluation results show that the 3D-FETO system is able to work around different kinds of hard faults and soft errors, ensuring graceful performance degradation, while minimizing additional hardware complexity and remaining power efficient.

This project is partially supported by Competitive Research Funding (CRF), The University of Aizu, Reference P-11 (2016), and JSPS KAKENHI Grant Number JP30453020.

Khanh N. Dang
d8162103@u-aizu.ac.jp

Abderazek Ben Abdallah
benab@u-aizu.ac.jp

1 Adaptive Systems Laboratory, Graduate School of Computer Science and Engineering, The University of Aizu, Aizu-Wakamatsu, Fukushima 965-8580, Japan
Keywords 3D NoCs · Fault-tolerance · Soft–hard faults · Reliability · Architecture · Design

1 Introduction

Global interconnects are becoming the principal performance bottleneck for high-performance Systems-on-Chip (SoCs) [2]. The 3-dimensional Networks-on-Chip (3D-NoCs) have been proposed as a promising architecture that combines the high parallelism of Network-on-Chip paradigm with the high performance and lower interconnect power of 3-dimensional integration circuits (3D-ICs) [6]. In the past few years, the benefits of 3D integrated circuits (3D-ICs) and mesh-based Network-on-Chips (NoCs) have been fused into a promising architecture opening a new horizon for IC design. The parallelism of NoCs can be enhanced in the third dimension thanks to the short wire length and low power consumption of the interconnects of 3D-ICs. As a result, the 3D-NoC paradigm is considered to be one of the most advanced and auspicious architectures for the future of IC design, as it is capable of providing extremely high bandwidth and low power interconnects.

While the NoC paradigm has been increasing in popularity with several commercial chips [3], it is threatened by the decreasing reliability of aggressively scaled transistors. Transistors are approaching the fundamental limits of scaling. Gate widths are nearing the molecular scale, resulting in breakdown and wear out in end products [19,23]. Moreover, the anticipated fabrication geometry in 2018 scales down to 8 nm with a projected 0.6 V supply voltage [22]. In the 8-nm process, a higher rate of soft errors affect control logic and buffers of NoC routers, leading to chip failure. In addition, the low supply voltage enforces a very narrow noise margin, which makes the architecture vulnerable and sensitive to faults. As reported in [16], the soft error rate increases about 30% for each 100 mV decrease in the supply voltage. With rising power density and non-ideal threshold and supply voltage scaling, soft errors have become increasingly common during a chip’s lifetime [17]. Figure 1 shows a detailed taxonomy of different types of error and fault sources in NoCs. We categorized the faults into two classes: hard faults and soft errors.

![Fig. 1 Taxonomy of errors and faults in NoCs](image-url)
Hard faults, including both permanent faults and intermittent faults, can occur during the manufacturing stage or under specific operating circumstances. Intermittent faults periodically occur during operation and can disappear after a certain time. Because these faults do not permanently damage a given component, it can pass through several testing stages, but can still cause operation failures. Although intermittent faults can disappear after a specific period of time, their inconsistency can be treated as permanent faults to avoid complex situations. For both permanent and intermittent faults, the most natural solution is using redundant components [12,15].

Soft errors arise from energetic particles, such as alpha particles and neutrons from cosmic rays, generating electron–hole pairs as they pass through a device. A sufficient amount of accumulated charge may invert the state of a logic device such as a latch, gate, or SRAM cell, thereby introducing a logic fault into the NoC’s operation. Soft errors do not permanently defect the gate and only occur over a short period of time. Because of their special characteristics, they are unpredictable and unavoidable. Unlike permanent and intermittent faults, transient faults cannot be fixed by replacing the affected components. Instead, they can be recovered by repeating the erroneous operation. A transient failure inside the data path can also be fixed by using code-based techniques (e.g., error correction code (ECC) [8]). Statistically, transient faults are the most common kind of fault accounting for 80% of failures, as reported in [24]. Therefore, without an efficient protection mechanism, these errors can compromise the system’s functionality and reliability.

Hard fault handling schemes are based on two main approaches: (a) fault-tolerant routing algorithms, which enable packets to avoid faulty nodes in the network [6,15]; (b) architecture-based methods, which use hardware (components) redundancy and/or reconfiguration to recover from faults [1,12,15]. Soft error recovery is also solved by two main schemes: (a) data corruption handling using ECC-based methods [8,26,38]; (b) control logic handling using temporal redundancy-based methods [14,18,39].

Although many researchers have proposed solutions for various individual aspects of on-chip reliability, a comprehensive approach encompassing both soft errors and hard faults pertaining to NoC reliability has yet to evolve. In addition, the error detection and diagnosis in NoC architectures has been studied thoroughly in the scope of offline testing; however, with soft errors and intermittent faults becoming a dominant failure mode in modern NoCs and general VLSI systems, a widespread deployment of online test approaches has become crucial. In this paper, we present a comprehensive soft error and hard fault-tolerant 3D-NoC architecture, named 3D-Hard-Fault-Soft-Error-Tolerant-OASIS-NoC (3D-FETO). With the aid of efficient mechanisms and algorithms, 3D-FETO is capable of detecting and recovering from soft errors occurring in the routing pipeline stages and leverages reconfigurable components to handle permanent fault occurrences in links, input-buffers, and crossbars. The main contributions of this work are summarized as follows:

- A new adaptive 3D router architecture based on a robust hardware reconfiguration mechanism of the most susceptible components to hardware faults, and on a low-cost method that is capable of detecting and recovering from soft errors in the router pipeline stages.
An efficient scheme for online control fault detection and diagnosis in 3D-NoC systems.

The organization of this paper is as follows: in Sect. 2, we present related works. Section 3 presents the adaptive router architecture (SHER-3DR). In Sect. 4, we present comprehensive techniques which include fault detection, diagnosis and recovery. Section 5 provides the implementation and evaluation results. Finally, we present the conclusion and our ideas for future work in the last section.

2 Related works

Numerous works have addressed the fault-tolerance and reliability issues in NoC architectures. In [1, 6, 7], we covered some well-known solutions presented to tackle hard faults; therefore, in this section we mainly focus on solutions related to soft error recovery. As depicted in Table 1, they are classified into methods focusing on the data path (DP) and methods focusing on the control logic (CL) of the router.

For soft errors in the data path, most works use code-based techniques that not only detect the integrity of the received data, but also provide a correction function up to an acceptable number of faults. For instance, Bertozzi et al. [8] analyzed several low-cost coding techniques for on-chip communication. Among these techniques, single error correction and double error detection (SECDED) was found to be the solution with the most balanced trade-off between reliability and implementation cost. Although the authors provide several evaluations of energy and hardware complexity, on-chip communication analysis (such as throughput and latency) is missing. As an adaptive solution, Yu et al. [38] presented a dynamic ECC based on quality of wire connection by using a configurable ECC with two Hamming codes to adapt with several probabilities of faults. Although this adaptive ECC obtains energy efficiency, its area overhead is problematic.

Soft errors can be detected and recovered using temporal redundancy. For example, Ernst et al. [18] presented a Razor D Flip-Flop with an additional shadow latch sampled
by a delayed clock for checking the occurrence of transient faults. Furthermore, a soft error detection solution based on redundant latches was also presented by Ravindan et al. [34]. Although these techniques obtain more efficient detection results, they nearly double the area overhead and power consumption to maintain the redundant latches.

For soft errors in the control logic, there are several techniques with cross-layer resolution. In the End-to-End level, Shamshiri et al. [36] proposed error correction and online diagnosis using a specific code named $2G4L$. Based on the position of the erroneous bit in the received data, the system can indicate the position of the faulty node in the network; however, when a packet is misrouted due to wrong routing information/arbitration or an adaptive routing algorithm, the path of a packet is not fixed in a way that can determine the faulty node. To ensure arbitration computation across layers, NoCAalert [31] implements constraints to obtain computational accuracy. By constraining the relationship between the input and output of a block, the system can detect both soft and hard faults. Although this work presents efficient detection, it lacks efficiency in recovering from soft errors. First, the system needs to distinguish between soft and hard faults to decide the recovery method. Second, soft errors cannot be recovered by spatial redundancy and their recovery in the End-to-End level is inefficient. The FoReVer framework [29] also presented a network level method to detect and recover from routing errors: lost, duplicated, and misrouted packets. Since FoReVer is based on End-to-End detection and recovery, dealing with soft errors requires retransmission of the whole packet instead of an online recovery.

In the physical/data-link layers, one of the most common methods is using triple modular redundancy (TMR). By triplicating the original module, the system gets three results at the same time [32]. The three results are sent to a Majority Voting module to decide the accurate result. Although this technique suffers from high area overhead and power consumption (about 300%), it is easy to implement and effective for both soft errors and hard faults. In [39], the authors deploy a monitoring system on important control modules. They can diagnose the output to find the failure. This technique is light-weight in both area and power and has an insignificant impact on the system performance. However, it suffers from lack of flexibility since the monitor module has to be specifically designed depending on the target component. If any changes in the routing algorithm or pipeline stages are needed, investigation and re-designing of the monitor module is mandatory.

3 Adaptive 3D router architecture (SHER-3DR)

Figure 2 shows the block diagram of the proposed adaptive 3D router architecture (SHER-3DR). The router relies on simple recovery techniques based on system reconfiguration with redundant structural resources to contain hard faults in the input-buffers, crossbar, and links, in addition to soft errors in the routing pipeline stages.

The SHER-3DR router is the backbone component of the 3D-FETO system. Each router has a maximum of 7-input and 7-output ports, where 6 input/output ports are dedicated to the connection to the neighboring routers and one input/output port is used to connect the switch to the local computation tile. As shown in Fig. 2, the
SHER-3DR contains seven Input-port modules for each direction in addition to the Switch-Allocator, and the Crossbar module which handles the transfer of flits to the next node. An Input-port module is composed of two main elements: an Input-buffer and the LAFT routing (Next-Port-Computing) module. Incoming flits from different neighboring routers, or from the connected computation tile, are first stored in the Input-buffer. This step is considered to be the first pipeline stage of the flit’s life-cycle, Buffer-Writing (BW). After receiving and storing the flits, their routing information is read and processed by a LAFT-Routing module (Next-Port-Computing) and an arbitrating module (Switch-Allocator). This step is the second stage—Next-Port-Computing/Switch-Allocator (NPC/SA). After the NPC/SA pipeline stage, the next-port value is merged into the flit and the grant signal allows the flit to traverse from its input port to an output port (Crossbar-Traversal (CT) stage).

An augmented Look-Ahead-Fault-Tolerant routing algorithm (LAFT) [4,5] is used to perform the routing decision. If a given flit is routed to the local port, there is no routing calculation. If the flit is to be routed to another node, the fault link information of all neighboring nodes is read by each input-port and LAFT routing is executed. The first phase of the algorithm is calculating the next node’s address and its fault output information. In the next phase, the LAFT routing algorithm determines the minimal paths which are valid for routing after eliminating the faulty paths. The final routing path is selected by evaluating two factors of all the possible routing paths: (1) the diversity of the routing path to the destination node and (2) the congestion value of the connection. If there is no minimal routing path, a similar approach is applied for the non-minimal routing paths. Finally, an output port of the selected routing is calculated. This information is merged in the flit as next-output-port bits for routing in following nodes [1].
3.1 Hard fault recovery mechanism overview

The block diagram of the hard fault recovery mechanism is shown in Fig. 3. The Random Access Buffer mechanism (RAB) [1] solves the deadlock problem that can occur with the look-ahead fault-tolerant routing algorithm (LAFT), and is able to recover from transient, intermittent, and permanent faults in the input-buffer. When a fault is detected in one of the slots, the main controller (located in input port manager in Fig. 2) considers the flagged slots when assigning the write and read addresses. It remains to check the flagged slots for recovery from the faults.

The Bypass Link on Demand mechanism (BLoD) [1] provides additional escape channels whenever the number of faults in the baseline 7x7 crossbar increases. When a fault is detected in one or several crossbar links, the fault_manager (depicted in Fig. 2) disables the faulty crossbar links and enables the appropriate number of bypass channels. The number of bypass-links is very important and it should be minimized as much as possible to reduce the area and power overhead. In a case where the number of faulty links is larger than the number of backup links, the system needs to mark the links as faulty and use the LAFT algorithm to avoid routing through this defective connection.

3.2 Soft error recovery mechanism

As represented in Fig. 4, the principal soft-error handling method in the proposed 3D-FETO system relies on a solution called Pipeline Computation Redundancy (PCR) in one more clock cycle [14].

For ease of understanding, we explain the PCR in Algorithm 1. The Next Port Computing (NPC) and Switch Allocator (SA) run in parallel (lines 2, 3) after the Buffer Writing stage. This is achieved by the LAFT routing algorithm, where the
dependency between the two stages is eliminated. After the first computation, both of the stages have an additional computation clock cycle (lines 4, 5). By comparing two consecutive results, soft errors will be detected. If a soft error is detected, the whole pipeline is halted for correction. A third computation is required for majority voting, which decides the final result. To recover from soft errors in the data, Single Error
Correction Double Error Detection (SECDED) [21] with Automatic Retransmission Request (ARQ) [26] is adopted.

In the first stage, flits are stored in the input buffer at the Buffer Writing (BW) stage, and the ECC is used to check and correct the input data in the ECC module. In second stage, the NPC and the SA are executed in parallel in the LAFT routing unit and the Switch-Allocator module. In third stage, the Redundant NPC (RNPC) and the Redundant SA (RSA) are computed in parallel. Then, if the output of RNPC is equal to that of NPC, and SA is equal to RSA, the Crossbar Traversal (CT) stage is performed in the third cycle, and the flit goes to the next router via the output channel. If the RNPC is not equal to the NPC, the system rolls-back and recomputes the NPC. Moreover, if SA is not equal to RSA, the system also rolls-back and re-computes the SA stage. After rolling-back, re-computing, a majority voting module is used to decide the correct output of these modules. The rolling-back, re-computing and voting are executed. Then, the outputs of NPC/SA are sent to the Crossbar Traversal stage to finish the flit transmission.

Figure 5 presents a working demonstration of the SHER-3DR router. \([flit(n)]\) represents the flit in the \(n\)th position of the packet. \([time(m)]\) illustrates the \(m\)th time of computation. In the first clock cycle, BW handles \([flit(1)]\) while NPC/SA and
CT are idle or are handling another packet. In the second cycle, NPC/SA computes \([\text{flit}(1), \text{time}(1)]\), which means the computation of the first flit for the first time. In the third cycle, NPC/SA computes \([\text{flit}(1), \text{time}(2)]\), which means that it computes the first flit for the second time, also known as the redundant computation. \([c(1)]\) compares the results of \([\text{flit}(1), \text{time}(1)]\) and \([\text{flit}(1), \text{time}(2)]\) to detect the occurrence of a soft error. If there is no error, CT processes \([\text{flit}(1), \text{time}(1)]\) to finish the pipeline stages of the first flit. If there is an error in NPC/SA, the system requires the recovery in the fourth cycle. In this cycle, NPC/SA recalculates the first flit for the third time \([\text{flit}(1), \text{time}(3)]\) and finalizes an accurate result by using majority voting \([f(1)]\). After getting the final result of the first flit, CT completes the pipeline stage of the first flit based on the correct result of the two previous computations: \([\text{flit}(1), \text{time}(1)]\) or \([\text{flit}(1), \text{time}(2)]\). As shown in Fig. 5, the router requires one clock cycle for detecting a soft-error and one optional cycle for recovering each time an error occurs.

## 4 Light-weight detection, diagnosis and recovery mechanism (DDRM)

Algorithm 2 shows the proposed Detection, Diagnosis and Recovery Mechanism (DDRM). It uses the feedback from the ECC and the ARQ protocol to monitor the errors. As shown in Fig. 2, the input data is first verified by an ECC decoder. If the value is correct or the ECC decoder can handle the correction, the flit is written to the input buffer. Otherwise, a retransmission is requested. Since the transient fault only occurs over a short period of time, assumed to be a single clock cycle, it does not occur
for two consecutive cycles. Therefore, ARQ can recover this kind of fault. However, if a permanent fault occurs, ARQ is unable to correct it and the faulty connection will keep requesting retransmission infinitely. Therefore, if the ARQ cannot correct the fault, the system considers it to be a permanent fault (lines 1–10 in Algorithm 2).
Since a flit’s correctness is verified by the ECC module before being written to the buffer, a permanent fault can only occur in the path between the input-buffer in the upstream node and the one in the downstream node. Figure 6 shows the high-level view of the DDRM and router-to-router interfacing. The transmission path of a flit consists of three main components: input buffer slots, a crossbar link and a router-to-router channel. When a fault is detected, DDRM diagnoses these two components to find the fault position and recover it with an appropriate mechanism.

For the diagnosis and recovery phase, the router’s Fault-manager module initiates the diagnosis with input buffer checking. In this step, the error statuses of the following flits of the monitored input buffer are checked. If errors are detected in the following flits’ transmission, it means the fault should belong to the crossbar link or the inter-router channel. The diagnosis is forwarded to check the crossbar and inter-router channel. If errors are constantly detected at the same position of the monitored buffer, the fault belongs to this detected position. In this fashion, the Fault-manager sends a signal to the RAB mechanism to indicate the faultiness of the slot in the input buffer (lines 11–14). If the fault-manager indicates that the fault may belong to the crossbar or inter-router channel, the Fault-manager first configures the Bypass-Link-on-Demand (previously presented in Sect. 3.1) to establish an alternative connection path. Then, another flit is sent from the input buffer through a bypass-link and the router-to-router channel to the downstream node. If, at the downstream node, the flit is found to be not faulty by the ECC module, the Fault-manager concludes that the fault is in the crossbar, which is already handled by the BLoD mechanism. Therefore, the configuration of the BLoD is kept as a recovery. If the flit is still faulty, the fault belongs to the inter-router channel. In this situation, the BLoD is released for further fault-tolerance and the information of the faulty channel is sent to the routing module (in LAFT algorithm). At the routing module, the Look-Ahead Fault-Tolerant routing algorithm uses the fault information to handle the channel’s failure. The flit in the input buffer is re-routed via an alternative output port.
5 Evaluation results

5.1 Evaluation methodology

The proposed 3D-FETO system was designed in Verilog-HDL, synthesized and prototyped with commercial CAD tools and VLSI technology, respectively [27, 28]. We evaluate the hardware complexity of the SHER-3DR router in terms of area utilization, power consumption (static and dynamic), and speed. To evaluate the performance of the proposed system, we select both synthetic and realistic traffic patterns as benchmarks. For synthetic benchmarks, we selected Transpose [11], Uniform [37], Matrix-multiplication [10, 40], and Hotspot 10% [13]. For realistic benchmarks, we chose H.264 video encoding system [33], Video Object Plane Decoder (VOPD), Picture In Picture (PIP) and Multiple Window Display (MWD) [9]. The simulation configurations are depicted in Table 2.

The above synthetic benchmarks help us understand the performance of the network under stress; however, we also need several realistic benchmarks to understand the network under real application traffic. Therefore, we build a simulator in Verilog-HDL which allows us to set up the traffic patterns from real applications. Based on the traffic patterns, the Network Interfaces send and receive packets over the networks. We select a video encoding system using a H.264 encoder, a MP3 encoder, and a OFDM [33]. Moreover, we select three applications [9]: VOPD, PIP and MWD.

We evaluate the performance of our fault-tolerant model which includes hard fault tolerance from 3D-FTO [1], soft-error tolerance OASIS system, and the proposed system (3D-FETO). We measure the average packet latency, with the selected synthetic and realistic benchmarks. To understand the impact of fault-tolerance techniques on performance, we compare the obtained results with the baseline 3D-NoC system presented in [4]. We randomly inject faults at three fault rates: 10, 20 and 33%. The faults are injected into hard fault-tolerant and soft error-tolerant modules. For the soft error-tolerant system, only soft errors are injected. For the hard fault-tolerant (3D-FTO) system, only hard faults are injected. For the final system (3D-FETO), both soft errors and hard faults are injected. Hard faults are injected at the beginning of simulation and their rate is measured as the percentage of routers with faults. Soft errors are injected during the system’s operation and their rate is considered to be the number of soft errors per clock cycle. The injected fault rates are considered individually for each error type.

5.2 Complexity evaluation

In this evaluation, we considered the hardware complexity of the proposed SHER-3DR router. For this evaluation, we use the NANGATE 45 nm technology library [27]. Area cost and power consumption analyses are performed with the Synopsys © Design Compiler. The power consumption information is analyzed based on the switching activity of the router under the uniform benchmark. We start first by observing the additional hardware added to the baseline system when we employ the hard fault tolerance model (3D-FTO router). Then, we evaluate the impact when we con-
sider the soft error-tolerant model (soft error-tolerant router). Finally, we evaluate the completed SHER-3DR system including both soft and hard fault-tolerant mechanisms. The configurations of the network are shown in Table 2 and the layout of a single SHER-3DR router is depicted in Fig. 7.

Table 3 illustrates the hardware complexity results of SHER-3DR router in terms of area, power (static, dynamic, and total), and speed. In the hard fault tolerance router (3D-FTO), the area and power consumption overheads have increased by 1.43 and 25.65%, respectively. The maximum speed has also slightly decreased. On the other hand, our soft error handling mechanism adds seven ARQ buffers and some combinational logic which increase the area and power consumption more significantly. However, SHER-3DR introduces 7.50 and 3.74% extra area and power consumption,

| Parameter/System | Value |
|------------------|-------|
| Network Size ($x \times y \times z$) | Matrix $6 \times 6 \times 3$ |
| | Transpose $4 \times 4 \times 4$ |
| | Uniform $4 \times 4 \times 4$ |
| | Hotspot 10% $4 \times 4 \times 4$ |
| | H.264 $3 \times 3 \times 3$ |
| | VOPD $3 \times 2 \times 2$ |
| | MWD $2 \times 2 \times 3$ |
| | PIP $2 \times 2 \times 2$ |
| Total Injected Packets | Matrix 1,080 |
| | Transpose 640 |
| | Uniform 8,192 |
| | Hotspot 10% 8,192 |
| | H.264 8,400 |
| | VOPD 3,494 |
| | MWD 1,120 |
| | PIP 512 |
| Packet’s Size | Hotspot 10% 10 flits + 10% for hotspot nodes |
| | Others 10 flits |
| Flits Size | 44 bits |
| Header Size | 14 bits |
| Payload Bit | Baseline, 3D-FTO 30 bits |
| | Soft Error Tolerance, 3D-FETO 18 bits |
| Parity Bit | Baseline, 3D-FTO 0 bits |
| | Soft Error Tolerance, 3D-FETO 12 bits (2 × SECDED(22,16)) |
| Buffer Depth | 4 |
| Switching | Wormhole-like |
| Flow-control | Stop-Go |
| Routing | LAFT |
A low-overhead soft–hard fault-tolerant

Fig. 7 Layout of a single SHER-3DR router for the 3D-FETO system. The SHER-3DR router was designed in Verilog-HDL and synthesized using 45 nm technology library [27]. For the Through Silicon Via (TSV) integration, we used FreePDK3D45 kit compiler [28]. The SHER-3DR router is designed on a 450 µm × 450 µm and the TSV array is 208 TSVs respectively, when compared to the soft error-tolerant model. In comparison to the baseline model, SHER-3DR increases the area and power consumption by 56.39 and 112.10%, respectively, while the maximum speed decreases by 33.70%.

The area cost and power consumption of the proposed router is given by Eq. 1, where \( \pi_i \) represents the area cost or power consumption of module \( i \). The SHER-3DR router consists of four main modules: input-ports, switch-allocator, crossbar, and fault manager.

\[
\pi_{\text{router}} = \pi_{\text{input-ports}} + \pi_{\text{switch-allocator}} + \pi_{\text{crossbar}} + \pi_{\text{fault-manager}} \tag{1}
\]

The details of an input port, a switch-allocator and a crossbar are given in Eq. 2.

\[
\begin{align*}
\pi_{\text{input-ports}} &= \pi_{\text{original-input-ports}} + \pi_{\text{RAB-controller}} \\
&\quad + \pi_{\text{PCR-controller}} + \pi_{\text{ECC}} \\
\pi_{\text{switch-allocator}} &= \pi_{\text{original-switch-allocator}} + \pi_{\text{PCR-monitor}} \\
\pi_{\text{crossbar}} &= \pi_{\text{original-crossbar}} + \pi_{\text{bypass-links}} + \pi_{\text{ARQ-buffers}} \tag{2}
\end{align*}
\]
We can observe the overheads in power consumption and area cost that are caused by the fault-tolerance mechanisms (RAB-controller, PCR-controller, ECC, BLoD, ARQ buffers). Figure 8 provides the evaluation results of power consumption and area cost of SHER-3DR. In terms of area cost, the input ports occupy the majority with over 67% which is followed by the crossbar (20%) and the switch allocator (9%). The fault manager, which supports DDRM, uses only about 4% of the overall area cost. In terms of power consumption, the input ports consume over 80% of the total value. The fault manager module also causes an insignificant increase in power consumption (3%).

When compared to the baseline OASIS router, the proposed SHER-3DR consumes more power consumption and costs more area. As shown in Fig. 8, SHER-3DER increases the area and power of all three main modules (crossbar, input ports, and switch-allocator). The overhead can be analyzed by Eq. 2 where additional modules are attached to support the fault-tolerance mechanisms.

Although our proposed models are penalized in terms of area, power consumption, and maximum frequency due to additional logic and registers that are necessary for fault handling mechanisms, they provide an improved resiliency against a significant amount of soft and hard faults.

### 5.3 Latency evaluation

In the second experiment, we evaluate the performance of the proposed architecture in terms of latency over various benchmark programs and error injection rates for three system configurations: (1) hard fault-tolerant system (3D-FTO); (2) soft error-tolerant OASIS system, and (3) hard fault and soft error-tolerant system (3D-FETO). The simulation results are shown in Figs. 9 and 10. From these graphs, we notice that with 0% hard faults (in input buffer and crossbar only), 3D-FTO has similar performance to the baseline system (LAFT-OASIS). In addition, we found that even
A low-overhead soft–hard fault-tolerant… 2721

Fig. 9  Average packet latency evaluation of the synthetic benchmarks

at a 33% fault rate, 3D-FTO increases the latency by only 1.71, 11.38, 8.79 and 13.73% for Transpose, Uniform, 6 × 6 Matrix, and Hotspot-10%, respectively. With realistic benchmarks, the performance of 3D-FTO slightly degrades at low error rates, but it suffers more of an impact at high error rates (20 and 33%) since the flit encounter bottlenecks due to errors inside the input buffers. However, the proposed 3D-FETO model still works even at high fault rates while the baseline model collapses at a 5% error rate. We used the same benchmark programs to evaluate the soft error-tolerant model. Since both the proposed Pipeline Computation Redundancy mechanism and ECC require additional clock cycles, we can observe a significant effect on average packet latency. For the 0, 10, 20 and 33% fault rates, the soft error-tolerant model increases the average delay in the Transpose benchmark by 18.57, 28.74, 34.54 and 49.62%, respectively. Finally, we evaluate the proposed 3D-FETO system with both soft error and hard fault handling schemes. As shown in Figs. 9 and 10, 3D-FETO has demonstrated a significant impact on the average latency, which has mostly doubled for both realistic and synthetic benchmarks. At a 33% fault rate using Matrix, Uniform, Transpose benchmarks, 3D-FETO’s average latency increases by 78.44, 50.73 and 67.18% in terms of average packet latency. The degradation is caused by both soft errors and hard fault tolerance mechanisms: (1) the ECC+ARQ and PCR both require additional re-transmission clock cycles; (2) the RAB and LAFT routing algorithm may disable a part of the network which causes congestion. However, it still maintains
the ability to work under an extremely high fault rate (33% for hard faults and 33% for soft errors).

5.4 Throughput evaluation

Figure 11 depicts the throughput evaluation with the adopted synthetic benchmarks. At a 0% error rate, 3D-FTO (hard-fault tolerance) presents the best throughput which matches the capacity of the baseline LAFT-OASIS. The soft error-tolerant OASIS and the proposed 3D-FETO have less throughput due to their soft error tolerance mechanisms. When the errors are injected into the system, we can observe a degradation in throughput. Thanks to the efficient hard fault tolerance scheme and the fault-tolerant routing algorithm, 3D-FTO at a 33% error rate provides a slightly decreased throughput: 40.18, 43.96, 43.55 and 32.59% for Transpose, Matrix, Uniform, and Hotspot 10%, respectively. For the soft error-tolerant OASIS, the system requires re-transmission via the ARQ mechanism and the re-execution for the soft error mechanism. Therefore, the throughput is degraded due to extra clock cycles. The proposed 3D-FETO, which is a fusion of both hard fault tolerance and soft error-tolerant mechanisms, inherits both degradations; however, these systems provide the ability to handle up to a 33% error rate (the limitation of the soft error mechanism).
5.5 Reliability evaluation

5.5.1 Arrival rate

This subsection presents the reliability evaluation of the proposed 3D-FETO system over several hard fault and soft errors injection rates. For comparison, seven systems adopting different routing algorithms are selected [30]: XYZ, Hybrid-XYZ, 8-Random-Walk (8-RW), Odd-Even, Hybrid-Odd-Even, 4N-First, and 4NPF-First. Among these algorithms, we can find deterministic 3D routing algorithms, fault-tolerant 2D algorithms that were extended to the third dimension, and also turn-model-based schemes that were proposed for fault-tolerant 3D-NoC systems. We adopted the same simulation environment and assumptions made in [30] from where the arrival-rate results were also obtained. For fair comparison, we assume that the faults can occur at any link with LAFT; thus, we eliminate the two assumptions that are necessary for the algorithm to efficiently work: (1) the links connecting the PE to the local input and output ports are always non-faulty. (2) There exists at least one non-faulty path between a (source, destination) pair. Moreover, we also evaluate the arrival rate of our final system with the enhancements by Random-Access-Buffer and Bypass-Link-on-Demand. Instead of only distributing faults on the inter-router channel, they are randomly assigned to input buffers, crossbar, or the inter-router channel.
Table 4 Successful arrival-rate comparison results for a $5 \times 5 \times 4$ system configuration under Uniform traffic

| Algorithm/fault rate (%) | 1  | 5  | 10 | 15 | 20 |
|--------------------------|----|----|----|----|----|
| XYZ (%)                  | 91 | 62 | 41 | 28 | 23 |
| Hybrid-XYZ (%)           | 99 | 83 | 62 | 44 | 36 |
| 8-RW (%)                 | 100| 95 | 85 | 69 | 59 |
| Odd-Even (%)             | 96 | 85 | 67 | 53 | 43 |
| Hybrid-Odd-Even (%)      | 100| 94 | 83 | 70 | 61 |
| 4N-FIRST (%)             | 98 | 89 | 72 | 68 | 46 |
| 4NP-FIRST (%)            | 97 | 98 | 95 | 83 | 76 |
| LAFT-OASIS (%)           | 100| 100| 99 | 98 | 95 |
| 3D-FETO (%)              | 100| 100| 99 | 99 | 97 |

Table 5 Successful arrival-rate comparison results for a $5 \times 5 \times 4$ system configuration under Transpose traffic

| Algorithm/fault rate (%) | 1  | 5  | 10 | 15 | 20 |
|--------------------------|----|----|----|----|----|
| XYZ (%)                  | 85 | 46 | 31 | 14 | 11 |
| Hybrid-XYZ (%)           | 99 | 68 | 42 | 25 | 20 |
| 8-RW (%)                 | 93 | 82 | 62 | 44 | 36 |
| Odd-Even (%)             | 97 | 84 | 53 | 42 | 32 |
| Hybrid-Odd-Even (%)      | 99 | 92 | 77 | 62 | 53 |
| 4N-FIRST (%)             | 96 | 86 | 68 | 50 | 37 |
| 4NP-FIRST (%)            | 100| 97 | 89 | 75 | 63 |
| LAFT-OASIS (%)           | 100| 100| 100| 99 | 96 |
| 3D-FETO (%)              | 100| 100| 100| 99 | 98 |

Tables 4 and 5 depict the arrival-rate results for a $5 \times 5 \times 4$ system (100 nodes) under Uniform and Transpose traffic patterns, respectively. Due to its lack of support for fault-tolerance, XYZ routing demonstrates the worst arrival rate for both applications. Its variant Hybrid-XYZ shows slightly better results, but it is still considered unacceptable. Despite the fact that 8-RW is fault-tolerant, its arrival rate considerably degrades as we increase the fault rate. This can be explained by the frequent deadlock-occurrence with this algorithm that is considered to be one of its main drawbacks. 4NP-FIRST is a fault-tolerant routing algorithm targeted for 3D-NoCs. However, it does not scale very well when we increase the fault rate. In fact, one-third of the injected packets fail to reach their destinations at a 20% fault rate, which can be seen in Table 5.

Among the considered algorithms, 3D-FETO appears to be the most reliable solution, providing a scalable arrival rate that does not go under 97% in both applications, even at a 20% fault rate. When observing the results with the two applications, 3D-FETO with the LAFT algorithm is considered to be the only scheme that takes advantage of long distance communications in Transpose traffic. This is in contrast with the remaining algorithms where their reliability degrades considerably with this application. In fact, the combination of look-ahead routing and the path prioritization using the diversity value in LAFT significantly increases the probability for packets to find non-faulty paths to reach their destinations.
The arrival rates of the proposed 3D-FETO reach over 97% in the worst case (20% fault rate) while LAFT-OASIS’s arrival rates are 95 and 96%. With other rates, 3D-FETO presents its capacity for high reliability with an arrival rate of over 98%. When we analyzed the possible causes for the failing 5%, we observed the occurrence of cases where all the connecting links of a given router are faulty: for example, the East, North, and UP links of the bottom-left router of the network are broken. Thus, the router cannot receive or inject any flit from/to the network. Another failure case manifests when the link connecting the router to the attached PE is faulty. As expected, these two cases justify the two assumptions that we previously made to ensure the efficiency of LAFT’s fault-tolerance capabilities.

5.5.2 Mean Time To Failure improvement

Besides the arrival rate evaluation, we assessed our fault-tolerant system in terms of Mean Time To Failure (MTTF) improvement. We define a system as healthy if it operates correctly (100% arrival rate, accurate fault detection and recovery function). Otherwise, the system is marked as failed. To obtain more precise results, we use the net-list (gate-level) models from the complexity evaluation. Moreover, faults are not only injected to the fault-tolerance modules, but they are also injected to other modules (controller, management module). Before the MTTF assessment, we first assume the original system has a natural fault rate: $\lambda_{raw}$. The MTTF value can be given as the following:

$$MTTF_{raw} = \frac{1}{\lambda_{raw}} \quad (3)$$

To measure the MTTF value of the fault-tolerant system, we use a Monte Carlo-based simulation as shown in Fig. 12. At the beginning of the simulation, we define the number of experiments ($N$) and the fault models and distribution mechanisms. Faults will be generated in two types: soft errors (randomly occur within a clock period) and hard faults (occur from the beginning to the end of experiment). There are also two fault models: stuck-at “0” and stuck-at “1”. Faults are injected to the dedicated gates which selected by a random generator. We use two distributions: (1) flat, randomly inject to any gate inside a router; (2) weight, more than 80% of faults are injected to the fault-tolerant modules (buffer, crossbar, next-port-computing, switch-allocator). For each experiment $i$, we inject faults and examine the correctness of the system (data’s accuracy, fault-tolerance configurations). Faults will be injected until the system is determined as failure. At the end of an experiment, the number of faults is recorded for the final process. To calculate the MTTF value of a system, the average number of faults is used in the following equation:

$$MTTF_{system} = \frac{\sum f_i \times MTTF_{raw}}{N} \quad (4)$$

In order to understand the efficiency of the fault-tolerance, the ratio of two MTTF values is used as in Eq. 5.
Define the total number of experiments (N)
Identify random parameters of the system
Assume appropriate distributions for the parameters

Initialize counter \( i = 1 \)

Generate a uniformly distributed number for each “experiment” \( i \)

Generate the random variable numbers to the system’s distribution

Evaluate by using the set of random number

Calculate the system MTTF:
\[
MTTF = \frac{\sum f_i \times MTTF_{raw}}{N}
\]

Is \( i = N? \)

No \( i = i + 1 \)

Yes

\( f_i = f_i + 1 \)

 succès

failure

Fig. 12 MTTF simulation methodology

| Fault-type   | Distribution | Baseline router | SHER-3DR router | MTTF improvement |
|--------------|--------------|-----------------|-----------------|------------------|
| Hard fault   | Flat         | 2.37            | 4.58            | 1.93             |
|              | Weighted     | 2.055           | 6.085           | 2.96             |
| Soft error   | Flat         | 17.928          | 26.770          | 1.49             |
|              | Weight       | 4.037           | 21.492          | 5.32             |

Because the raw fault rate depends on the technology parameters and the operating conditions, they will require a highly complex evaluation. To alleviate the complexity, we assume the fault-tolerant and original system have a similar raw fault rate. Therefore, the MTTF improvement can be obtained by Eq. 6, where \( AFTF \) is average fault to failure.

\[
\text{Improvement}_{MTTF} = \frac{MTTF_{fault-tolerant}}{MTTF_{original}}
\]

\[
\text{Improvement}_{MTTF} = \frac{AFTF_{fault-tolerant}}{AFTF_{original}}
\]

Table 6 shows the average number of faults to failure after 1000 simulations. The test scheme is built to functionally verify the data communication and the fault-tolerance mechanisms. In the flat distribution, the proposed SHER-3DR enhances the MTTF of hard faults and soft errors by 1.93 and 1.49 times, respectively. With the weight distribution, the proposal shows more improvement since the faults focus on the fault-
tolerant modules. SHER-3DR’s hard fault tolerance is 2.96 times better the baseline OASIS router. In terms of soft error MTTF, SHER-3DR is 5.32 times better than the original router. In conclusion, we observe a significant improvement in terms of MTTF from our proposed mechanism. Along with the high arrival rates, we demonstrated the reliability enhancement of our system.

6 Conclusion and future work

In this paper, we proposed a comprehensive fault-tolerant 3D-Network-on-Chip (3D-NoC) system architecture for highly reliable many-core Systems-on-Chips (SoCs), named 3D-FETO. The proposed system is based on two approaches. First, a comprehensive mechanism to handle both soft error and hard faults in a 3D-NoC router is proposed. The hard fault support is achieved by leveraging reconfigurable components to handle permanent faults in links, input buffers, and crossbars, while soft error tolerance is obtained via efficient and light-weight software redundancy that enables fault recovery in the router pipeline stages. In the second approach, the system can support a detection, diagnosis and recovery technique which makes it independent of any complex and costly testing mechanisms commonly found in conventional systems.

Through extensive evaluation, we showed that the proposed 3D-FETO was able to recover efficiently from a significant number of soft and hard errors at different fault rates, reaching up to 33%. This means that 3D-FETO can provide up to a 98% packet arrival rate even when almost one-third of its components have failed. Despite the performance degradation and hardware complexity penalty, we still consider that this overhead is acceptable. This is because we made sure that the system is still functional at high fault rates where previously proposed systems fail to deliver packets. As reliability constitutes one of the main challenges in future SoC design, we demonstrated that the proposed 3D-FETO can be used as a reliable and independent system capable of ensuring fault resiliency in worst-case scenarios and that it can be adopted for mission-critical applications where correct data delivery is primordial.

As a future work, we are planning to investigate the faults within Through-Silicon-Vias of 3D-ICs/3D-NoCs to provide a sufficient fault-tolerance method for 3D-NoC systems. Moreover, the degradation factors of the reliability, such as thermal stress, operating voltages, design characteristics should be also studied.

Acknowledgements This work is partially supported by Competitive Research Funding (CRF), The University of Aizu, Reference P-11 (2016), and JSPS KAKENHI Grant Number JP30453020. This work is also supported by VLSI Design and Education Center (VDEC), the University of Tokyo, Japan, in Collaboration with Synopsys, Inc. and Cadence Design Systems, Inc. The first and the last authors in the author list are the main contributors of this work.

References

1. Ahmed AB, Abdallah AB (2016) Adaptive fault-tolerant architecture and routing algorithm for reliable many-core 3D-NoC systems. J Parallel Distrib Comput 93–94:30–43
2. Ben Abdallah A (2013) Multicore systems-on-chip: practical hardware/software design, 2nd edn. Atlantis, Karachi
3. Ben Abdallah A, Masahiro S (2006) Basic network-on-chip interconnection for future gigascale MCSoCs applications: communication and computation orthogonalization. In: Proceedings of the symposium on science, society, and technology (JASSST2006), pp 1–7
4. Ben Ahmed A, Ben Abdallah A (2012) LA-XYZ: low latency, high throughput look-ahead routing algorithm for 3D network-on-chip (3D-NoC) architecture. In: IEEE 6th international symposium on embedded multicore socs (MCSOC). IEEE, New York, pp 167–174
5. Ben Ahmed A, Ben Abdallah A (2012) Low-overhead routing algorithm for 3D network-on-chip. In: Third International Conference on Networking and Computing (ICNC), pp 23–32
6. Ben Ahmed A, Ben Abdallah A (2013) Architecture and design of high-throughput, low-latency, and fault-tolerant routing algorithm for 3D-network-on-chip (3D-NoC). J Supercomput 66(3):1507–1532
7. Ben Ahmed A, Ben Abdallah A (2014) Graceful deadlock-free fault-tolerant routing algorithm for 3D network-on-chip architectures. J Parallel Distrib Comput 74(4):2229–2240
8. Bertozzi D, Benini L, De Micheli G (2005) Error control schemes for on-chip communication links: the energy-reliability tradeoff. IEEE Trans Comput-Aided Des Integr Circ Syst 24(6):818–831
9. Bertozzi D, Jalabert A, Murali S, Tamhankar R, Stergiou S, Benini L, De Micheli G (2005) NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. IEEE Trans Parallel Distrib Syst 16(2):113–129
10. Chen P, Dai K, Wu D, Rao J, Zou X (2010) The parallel algorithm implementation of matrix multiplication based on ESCA. In: IEEE Asia Pacific Conference on Circuits and Systems (APCCAS). IEEE, New York, pp 1091–1094
11. Chien AA, Kim JH (1995) Planar-adaptive routing: low-cost adaptive networks for multiprocessors. J ACM (JACM) 42(1):91–123
12. Constantinides K, Plaza S, Blome J, Zhang B, Bertacco V, Mahlke S, Austin T, Orshansky M (2006) Bulletproof: a defect-tolerant CMP switch architecture. In: In: The twelfth international symposium on high-performance computer architecture. IEEE, New York, pp 5–16
13. Dally WJ, Towles BP (2004) Principles and practices of interconnection networks. Elsevier, Amsterdam
14. Dang KN, Meyer M, Okuyama Y, Tran XT, Ben Abdallah A (2015) A soft-error resilient 3d network-on-chip router. In: IEEE 7th International Conference on Awareness Science and Technology (iCAST), pp 84–90
15. DeOrio A, Fick D, Bertacco V, Sylvester D, Blaauw D, Hu J, Chen G (2012) A reliable routing architecture and algorithm for NoCs. IEEE Trans Comput-Aided Des Integr Circ Syst 31(5):726–739
16. Dixit A, Wood A (2011) The impact of new technology on soft error rates. In: In: 2011 international reliability physics symposium, pp 5B.4.1–5B.4.7
17. Eghbal A, Yaghini PM, Bagherzadeh N, Khayambashi M (2015) Analytical fault tolerance assessment and metrics for TSV-based 3D network-on-chip. IEEE Trans Comput 64(12):3591–3604
18. Ernst D, Kim NS, Das S, Pant S, Rao R, Pham T, Ziesler C, Blaauw D, Austin T, Flautner K, et al. (2003) Razor: a low-power pipeline based on circuit-level timing speculation. In: In: Proceedings 36th annual IEEE/ACM international symposium on microarchitecture (MICRO-36). IEEE, New York, pp 7–18
19. Fick D, DeOrio A, Chen G, Bertacco V, Sylvester D, Blaauw D (2009) A highly resilient routing algorithm for fault-tolerant NoCs. In: In: 2009 Design, Automation Test in Europe Conference Exhibition, pp 21–26
20. Hernández C, Silla F, Santonja V, Duato J (2008) Dealing with variability in NoC links. In: In: 2nd workshop on diagnostic services in network-on-chips, pp 4–10
21. Hsiao MY (1970) A class of optimal minimum odd-weight-column sec-ded codes. IBM J Res Dev 14(4):395–401
22. ITRS: 2012 Edition Update Process Integration, Devices, and Structures. Tech. rep., The International Technology Roadmap for Semiconductor (2012). http://www.itrs2.net/2012-itr1s.html. Accessed 16 June 2016
23. Karl E, Blaauw D, Sylvester D, Mudge T (2006) Reliability modeling and management in dynamic microprocessor-based systems. In: In: Proceedings of the 43rd Annual Design Automation Conference. DAC ’06ACM, New York, pp 1057–1060
24. Lehtonen T, Liljeberg P, Plosila J (2007) Online reconfigurable self-timed links for fault tolerant NoC. VLSI Des 2007:1–13
25. Lehtonen T, Wolpert D, Liljeberg P, Plosila J, Ampadu P (2010) Self-adaptive system for addressing permanent errors in on-chip interconnects. IEEE Trans Very Large Scale Integr (VLSI) Syst 18(4):527–540
26. Lin S, Costello D, Miller M (1984) Automatic-repeat-request error-control schemes. IEEE Commun Mag 22(12):5–17
27. NanGate Inc.: Nangate Open Cell Library 45 nm. http://www.nangate.com/. Accessed 16 June 2016
28. NCSU Electronic Design Automation: FreePDK3D45 3D-IC process design kit. http://www.eda.ncsu.edu/wiki/FreePDK3D45:Contents. Accessed 16 June 2016
29. Parikh R, Bertacco V (2011) Formally enhanced runtime verification to ensure NoC functional correctness. In: Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture. MICRO-44 ACM, New York, pp 410–419
30. Pasricha S, Zou Y (2011) A low overhead fault tolerant routing scheme for 3D Networks-on-Chip. In: 12th International symposium on quality on-chip design (ISQED). IEEE, New York, pp 1–8
31. Prodromou A, Panteli A, Nicopoulos C, Sazeides Y (2012) NoCALert: an on-line and real-time fault detection mechanism for network-on-chip architectures. In: Proceedings of the 2012 45th annual IEEE/ACM international symposium on microarchitecture (MICRO), pp 60–71
32. Radetzki M, Feng C, Zhao X, Jantsch A (2013) Methods for fault tolerance in networks-on-chip. ACM Comput Surv (CSUR) 46(1):8
33. Rahmani AM, Vaddina KR, Latif K, Liljeberg P, Plosila J, Tenhunen H (2014) High-performance and fault-tolerant 3D noc-bus hybrid architecture using arb-net-based adaptive monitoring platform. IEEE Trans Comput 63(3):734–747
34. Ravindan DK (2009) Structural fault-tolerance on the NOC circuit level. Tech. rep. Institut fur Technische Informatik, Universitat Stuttgart
35. Shamshiri S, Cheng KT (2009) Yield and cost analysis of a reliable noc. In: 27th IEEE VLSI Test symposium. IEEE, New York, pp 173–178
36. Shamshiri S, Ghofrani AA, Cheng KT (2011) End-to-end error correction and online diagnosis for on-chip networks. In: IEEE International Test Conference (ITC). IEEE, New York, pp 1–10
37. Sivaram R (1992) Queuing delays for uniform and nonuniform traffic patterns in a MIN. ACM SIGSIM Simul Dig 22(1):17–27
38. Yu Q, Ampadu P (2010) Transient and permanent error co-management method for reliable networks-on-chip. In: Fourth ACM/IEEE international symposium on networks-on-chip (NOCS). IEEE, New York, pp 145–154
39. Yu Q, Zhang M, Ampadu P (2013) Addressing network-on-chip router transient errors with inherent information redundancy. ACM Trans Embed Comput Syst (TECS) 12(4):105:1–105:21
40. Zekri AS, Sedukhin SG (2006) The general matrix multiply-add operation on 2D torus. In: 20th international parallel and distributed processing symposium (IPDPS). IEEE, New York, pp 8–16