A Soft-Error Resilient Route Computation Unit for 3D Networks-on-Chips

Alexandre Coelho*, Amir Charif*†, Nacer-Eddine Zergainoh*, Juan Fraire*‡ and Raoul Velazco*
*Université Grenoble Alpes, CNRS, Grenoble INP, TIMA, F-38000 Grenoble, France
†Computing and Design Environment Laboratory, CEA LIST, F-91191 Gif-sur-Yvette, France
‡Universidad Nacional de Córdoba, CONICET, Córdoba, Argentina
{Alexandre.Coelho, Amir.Charif, Nacer-Eddine.Zergainoh, Juan.Fraire, Raoul.Velazco}@univ-grenoble-alpes.fr

***Abstract*—Three-dimensional Networks-on-Chips (3D-NoCs) have emerged as an alternative to further enhance the performance, functionality, and packaging density of 2D-NoCs. However, the increasing complexity of NoC routers, the continuous miniaturization of silicon technology, the lower operating voltages, and the higher operating frequencies have made the NoC increasingly vulnerable to soft errors. In particular, transient faults occurring in the route computation unit (RCU) can provoke misrouting which may lead to severe effects such as deadlocks or packet loss, corrupting the operation of the entire chip. By combining a reliable fault detection circuit leveraging circuit-level double-sampling, with a cost-effective rerouting mechanism, we develop a full fault-tolerance solution that can efficiently detect and correct such fatal errors before the affected packets leave the router. To validate the proposed solution, we also introduce a novel method for simulation-based fault-injection based on the NoC's gate-level netlist. Experimental results obtained from a partially and vertically connected 3D-NoC indicate that our solution can provide a high level of reliability in the presence of errors, at the expense of an area and power overhead of 4.1% and 6.8% respectively.**

***Index Terms*—Network-On-Chip, Soft-errors, Fault-injection, Router Micro-architecture, Multi-core System On Chip.**

I. Introduction

Network-on-Chip (NoC) architectures have emerged as a scalable solution to the bus-based interconnects challenges for future multi-core system-on-chip designs [1]. Nevertheless, the conventional planar integration of integrated circuits has limited floor-planning choices with increasing number of processing elements attached, limiting the potential performance of 2D-NoC architectures. In order to extend 2D-NoC capabilities, multiple layers of active devices can be integrated in a three-dimensional NoC (3D-NoC) allowing to reduce interconnect lengths and thus, improving the overall NoC performance [2].

On the other hand, as the feature sizes of integrated circuits decrease aggressively, combinational logics become more susceptible to transient faults [3]. Specifically, *soft-errors* provoked by defects, radiation particles, or cross-talk noise, were generally masked and thus disregarded in older technology. However, as operating voltages become lower and clocking frequency increases, the design of integrated circuits with technology node below $0.25\mu m$ requires of particular attention to soft-error effects [4]. Indeed, NoC routers are not the exception since they have several combinational logic elements. In fact, the probability of the occurrence of a soft-error is higher in 3D-NoC routers which are more complex than 2D-NoC routers because of the increasing number of connecting ports. Therefore, the study of 3D-NoC routers reliability towards soft-errors becomes mandatory for future large-scale integration of dependable system-on-chip.

Components inside a NoC router are typically structured into two interacting modules, the *control path* and the *data path* [5]. Soft-errors occurring in the data path can affect the data encoded in the packet. Fortunately, this type of fault is easy to detect and correct through existing error detecting and error correcting codes [6]. By contrast, faults in the control path are harder to detect and correct, and may leave the network in an inconsistent state, ultimately causing the entire chip to fail. One of the most critical sections in the NoC's control path is the Route Computation Unit (RCU), as it is the one responsible for selecting the next output port (i.e. direction) that a packet must take at every hop. If the RCU fails, packets may be forwarded to wrong outbound ports (i.e., *misrouting*) eventually leading to deadlocks (cyclic dependencies between packets) or packet loss.

In this paper, we propose to enhance the reliability of 3D-NoCs by detecting and correcting errors provoked by transient faults in the RCU. The primary characteristic of our method is the ability to reliably and quickly *detect* misrouting based on a combination of a fault tolerance technique called double-sampling, and a complementary illegal turn detection method based on existing signals. These two methods are combined into a specific hardware unit called *fault detection circuit*, which can make decisions to recover from faults. The second characteristic is the capability to *correct* misrouting either by the reuse of the route computation samples provided by double-sampling or by directly rerouting the in-transit packet. In order to validate the robustness of the proposed solution, we also introduce a novel method to simulate transient faults based on a NoC's gate-level netlist. A thorough evaluation on a partially vertically connected 3D-NoC is performed to demonstrate the increased resiliency, and to estimate the area and power overhead of the proposed RCU architecture.

This paper is organized as follows. Related works are reviewed in Section II. Next, the 3D router architecture is introduced in Section III. Detection and correction mechanisms in RCU are described in Section IV. The Fault injection procedure is presented in Section V. The evaluation through a fault-injection campaign, as well as the performance analysis in terms of area, power, reliability and latency are discussed in Section VI. Finally, the conclusions of the paper are drawn in Section VII.

II. Related Work

Previous works have used *spatial redundancy* (i.e., execute parallel routing calculations) to deal with faults provoked by soft errors in NoCs. For example, the authors in [7] proposed the BulletProof router that employs N-modular redundancy (NMR) techniques to provide fault tolerance. The work in [8] proposed a fault-resilient routing unit for NoCs based on of a single and simplified redundant computation unit operating side-by-side with the RCU. While the RCU supports a fully adaptive routing algorithm, the redundant unit only supports limited paths, but it is only activated when errors are detected. In general, spatial redundancy approaches are expensive, as they require more silicon area than the baseline router.

Another approach to fault tolerance in NoCs is based on *temporal redundancy* (i.e., repeat routing calculations). Authors in [9] applied temporal redundancy in fully-connected 3D-NoCs. The authors in [10], [11] propose a mechanism to detect faults based on illegal turns in the chosen packet path. The fault detection is done at the neighboring routers which repeat a simplified route computation using input parameters provided by the previous router. If the selected direction is valid, the packet handling process continues, if not, either a new route is calculated or the packet is dropped. In general, temporal redundancy solutions incur in increased packet processing time.

Hybrid spatial and temporal approaches were also explored. For example, authors in [12] proposed to borrow RCUs from neighboring input ports in the NoC router. In this case, three different route computations are performed for each new packet, to then compare the resulting values in order to detect possible faults. Some of these calculations might happen in parallel (spatial redundancy) or in serial (temporal redundancy) depending on the router load. Indeed, since all arriving packets need to wait for two other RCUs to be available, this method can add significant delay in networks with heavy traffic load.

Packet retransmission techniques were also proposed to trigger recalculations when necessary. In [13], the authors proposed a mechanism to detect and recover from transient faults through the analysis of the requested output port. When the RCU request an invalid output port, the router triggers a new routing computation to correct the error (i.e., rerouting). If the fault cannot be detected, the next router in the path will detect the fault and send a negative-acknowledgement (NACK) message to the previous router unit asking for re-calculation and retransmission. In a similar approach, the FoReVer framework [14] presents a method to detect and recover lost, duplicated, and misrouted packets from routing errors. Since FoReVer is based on End-to-End detection and recovery, dealing with soft errors requires retransmission of the whole packet. In general, retransmission-based recovery mechanisms require additional retransmission buffers.

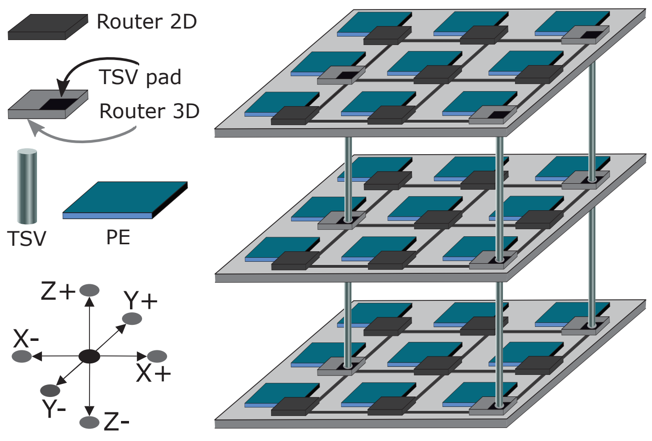

Fig. 1. 3D-NoC architecture

The work in [15] proposed to detect misrouting either in the faulty router or in the next-hop based on turns forbidden by the routing algorithm and blocked output ports (non-connected ports in the edge). The method is based on the fact that small components of the RCU are unlikely to evidence soft-errors allowing to simplify the correction. Thus, faults occurring at the address comparison stage are not detected. The primary correction mechanism is to reroute the packet. If faults are not detected in the faulty router, a dedicated network interface allows to re-inject the packet as a new packet in the NoC from the local router. Although re-injection minimizes packet drop, memory overhead is required to store the whole packet.

In this work, we propose to detect and correct *all* routing errors before the packet leaves the router. To achieve this goal, we make use of an additional fault detection circuit based on double-sampling and a rerouting technique to recover from soft-errors. Furthermore, and in contrast with previous 3D-NoC reliability works [9], we focus on the compatibility with partially vertically connected 3D-NoCs [16], [17].

III. 3D-NoC Architecture Overview

We consider a partially vertically connected 3D-NoC mesh architecture, wherein the routers include, in addition to the usual five ports (East, West, South, North, Local), either an Up port, a Down port, or both (i.e., 5, 6 and 7 port configuration). The Up and Down ports of the router are connected vertically using Through-Silicon Via (TSV), as shown in Figure 1. Each NoC router is identified by its coordinates (X, Y, Z), where X identifies the column, Y the row, and Z the layer.

The 3D-NoC routers used in this work consist of a five-stage pipeline: buffer write / route computation, virtual channel allocation, switch allocation, crossbar traversal and link traversal. When the header of a packet arrives at an input port, it is buffered in a FIFO virtual channel and, in the same cycle, the RCU reads the header information and calculates to which output port the packet should be forwarded. In the next cycle, the Virtual Channel Allocator (VCA) unit determines the virtual channel the packet can occupy in the downstream router. After VCA grant a virtual channel, the packet waits for the switch allocator unit to grant it permission to traverse the crossbar. Finally, the packet traverses the link to reach the next hop.

In order to prevent deadlocks, existing routing solutions for irregular 3D-NoCs employ virtual channels [18], [16]. In addition to the output port, the RCU also selects a virtual network (VN) number. This number is used by the VCA to determine the set of VCs that can be allocated to the packet as described in [16].

The route computation is performed in three stages [15]. In the first stage, the relative position of the packet's destination (Xd, Yd, Zd) is compared with the address of the current router (Xc, Yc, Zc) to produce a candidate direction vector [N', S', W', E', U', D']. In the direction vector, one or more signals may be active (e.g., if the destination is in the North-West quadrant, then N' and W' signals will be enabled). The second stage is used to add turn constraints according to the implemented adaptive routing algorithm. In this case, a vector [N", S", W", E", U", D", L"] representing the possible or legal directions is computed for each VN. Finally, in the third stage, an output port and a VN are selected among the legal routes by taking into account information such as congestion.

Since we consider that the route computation unit is responsible for selecting the output port and the VN, a transient fault in the RCU can leave an in-transit packet with an incorrect route, implying a wrong output port direction, an erroneous VN or both. In this case, to detect faults is required a mechanism capable of analyzing the output port and the VN selected by the RCU. Once detected, a correction mechanism need to be used to recover from the failure. It is worth mentioning that our solution can detect a fault affecting any of the three stages described above.

IV. Resilient Route Computation Unit

In this Section we will describe the techniques and mechanisms used to detect errors using double sampling, a custom VC allocator and novel correction procedures in the resilient RCU.

A. Detection: Double Sampling and Custom VC Allocator

The Double-Sampling (DS) is a method that permits to observe, at two different instants, the outputs of the combinational logic of each pipeline stage [19]. The main idea is to add a redundant sampling element (latch or flip-flop) to each output of the pipeline stages that need to be checked and clocking this redundant sampling element by a delayed clock signal.

Figure 2 illustrates the basic operation of the double-sampling mechanism used in this work. We propose to use the rising edge of the clock as latching event of the regular flip-flop and the falling edge of the clock as latching event of the redundant sampling element to get the output port and virtual network computed by the RCU ([OP,VN]). Since the RCU is performed in one clock cycle, and the double-sampling

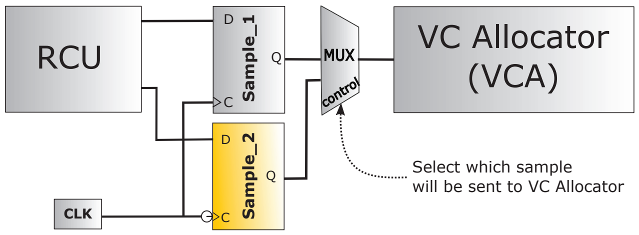

Fig. 2. A Double-Sampling implementation.

is running in parallel with RCU, this technique does not result in any path delay.

The Sample_1 and Sample_2 are obtained in different instants, as shown in Figure 2. Those two samples are compared by the additional Fault Detection Circuit (FDC) in the next cycle, after route computation. If the result of comparison between these two samples is different, then a fault has affected one of the samples. Therefore, both samples need to be analyzed with the purpose of determining which one is faulty. In our implementation, Sample_1 is initially verified, followed by Sample_2. On the other hand, if the result is equal, either there was not a fault, or a same fault affecting both samples. In this case, we only select Sample_1 to validate if there have been errors or not.

The detection of an incorrect route vector ([OP,VN]) can be performed directly by the VC Allocator (VCA) or by an extra Fault Detection Circuit (FDC), which runs in parallel with the VCA, as shown in Figure 3.

Because the VCA includes a table indicating the possible input-output connections according to the routing algorithm, it can detect illegal output port and VN requests with minor modifications. For example, if a request demanding for a blocked output port (i.e., the requested output port is unconnected because it is located at an edge of the 3D-NoC), the VCA can declare a routing error.

For faults that do not involve illegal turns, more sophisticated approaches are needed. This is the case when the output port and VN are both valid, but the packet needs to make illegal turns at later hops to reach its destination. To deal with these faults that cannot be detected by the VCA, an extra Fault Detection Circuit (FDC) is included in the NoC router, as shown in Figure 3. To this end, the FDC checks three conditions. First, it compares the Zd and Zc to know if the packets which need to go in the Up direction were

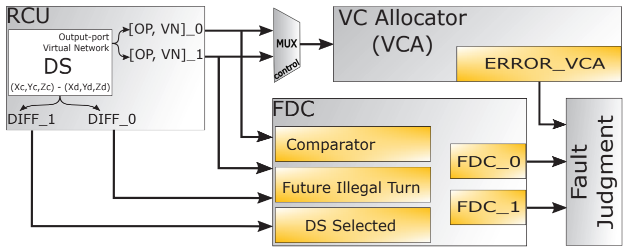

Fig. 3. Fault detection Circuit - FDC.

wrongly selected to go Down, or vice versa. Additionally, it checks if the packets are directed to a local port, in this case, the FDC compares the position of the current router with the destination of the packets. Second, the FDC checks if the direction adopted by the packet leaves to an illegal turn in the upcoming packet path. For example, if a routing unit using negative-first algorithm selects a valid direction, e.g. North which is a positive direction, but the destination is located West to the current router, this turn is declared illegal, as the packet will be forced to take the West (negative) direction at later hops after the North, leading to deadlocks. Third, it checks the two samples provided by the double-sampling and selects which of them must be finally used by the VCA.

B. Correction: Rerouting

The VCA and the FDC work together to detect transient faults in the RCU. The faults reported by the FDC and VCA are analyzed and judged as follows: If both samples (Samples_1 and Sample_2 from double-sampling) are the same and the ERROR_VCA output signalizes an error detected by the VCA, the solution will be asking for rerouting in the next clock cycle. On the other hand, if ERROR_VCA does not signalize an error, (i.e., the VCA was successful), the next steps will be to check both FDC_0 and FDC_1 signals (each signaling errors in each of the double samples) to declare a final judgment for the routed packet. If FDC_0 is fault-free, the result will be considered correct, and the packet will follow its path to the next router. Otherwise, if an error is detected in FDC_0, the next step will be to check the FDC_1 signal. If FDC_1 signalizes an error, then the decision will be to reroute the packet. However, if FDC_1 does not contain errors, the solution will be to try a new allocation in the VCA using the second sample (Sample_2). Finally, if the result is correct for this second sample, the packet will continue its path using this result. In all other cases, the packet will be rerouted.

Unfortunately, repeating the route computation process is not simple since one RCU is shared among all VCs in the same input port. This means that each port is able to do one route computation per clock cycle. To avoid replicating the RCU for each VC, we propose a simple rerouting mechanism that uses the existing RCU. The idea is to ensure that a limited number of new packets enter an input port in which one or more packets have requested rerouting. The solution consists in preventing packets from entirely leaving the input port, ensuring the buffers do not get available to receive new packets until all packets have been rerouted [15]. If a new packet header is received at the input port, it is routed by the RCU, otherwise one of the packets requesting rerouting is allowed to use the RCU to be rerouted. Because a finite number of new packets will be received, our method effectively guarantees that all requesting packets are eventually rerouted.

V. Fault-Injection Experimental Procedure

In order to analyze the reliability of the resilient RCU proposed in this work, we introduce a fault injection methodology that can mimic, with high accuracy, the effects of transient faults in the NoC's gate-level netlist. The most relevant characteristic of the resulting tool-chain is its ability to inject faults without modifying the original Hardware Description Language (HDL) code of the NoC. Instead of editing the original HDL design, the original library provided by the design kit used to make the synthesis of the design was modified. In particular, we have modified the design kit FD-SOI $28nm$ technology from ST-Microelectronics to inject faults in the NoC's gate-level netlist. Figure 4 illustrates the proposed work-flow.

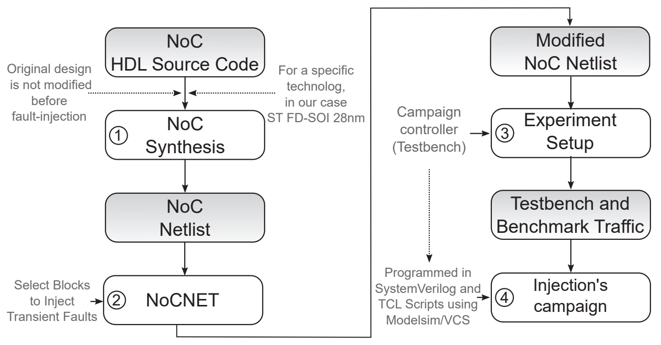
NoC HDL Source Code
Original design is not modified before fault-injection
For a specific technolog, in our case ST FD-SOI 28nm
① NoC Synthesis
NoC Netlist
Select Blocks to Inject Transient Faults
② NoCNET
Modified NoC Netlist
Campaign controller (Testbench)
③ Experiment Setup
Testbench and Benchmark Traffic
Programmed in SystemVerilog and TCL Scripts using Modelsim/VCS
④ Injection's campaign

Fig. 4. Fault injection methodology

Initially, the HDL description of the NoC is used to obtain a first synthesis in Step ①. As previously stated, this first step does not require any modification in the original design of the NoC. Although different tools can be used to accomplish this step, we have used Synopsys Design Compiler as it allows to export a gate netlist based on the technology adopted, in our case ST FD-SOI $28nm$.

In Step ②, the netlist is used as input for the NoC NETlist (NoCNET) tool. The output of NoCNET is a modified (but functionally equivalent) netlist with a large number of extra input signals used to access all logic blocks of the NoC to inject faults. The resulting synthesis of the modified netlist includes some additional combinational circuitry to the design. In the case of transient faults, NoCNET modifies all the logic gates of the netlist by simply adding an extra multiplexer at the output to select the appropriate value (erroneous or correct).

In Step ③, a campaign controller is integrated within the modified netlist. The campaign controller is a HDL testbench implemented in SystemVerilog that is in charge of managing the fault injection campaign by being directly wired to the NoC modified by NoCNET. To this end, the netlist obtained in Step ② is attached to the testbench by using the modified library (ST FD-SOI $28nm$).

Finally, the experiment in Step ④ can be directly executed using simulator tools like Modelsim (Mentor Graphics) or VCS (Synopsys). The whole fault-injection campaign (including the post-processing of the results), can be conveniently encoded in the testbench and automated by means of Tool Command Language (TCL) scripts. By accessing the interfaces connecting the NoC, fault injection signal and network interfaces, the testbench can efficiently execute several iterations of fault-injection experiments randomly selecting the

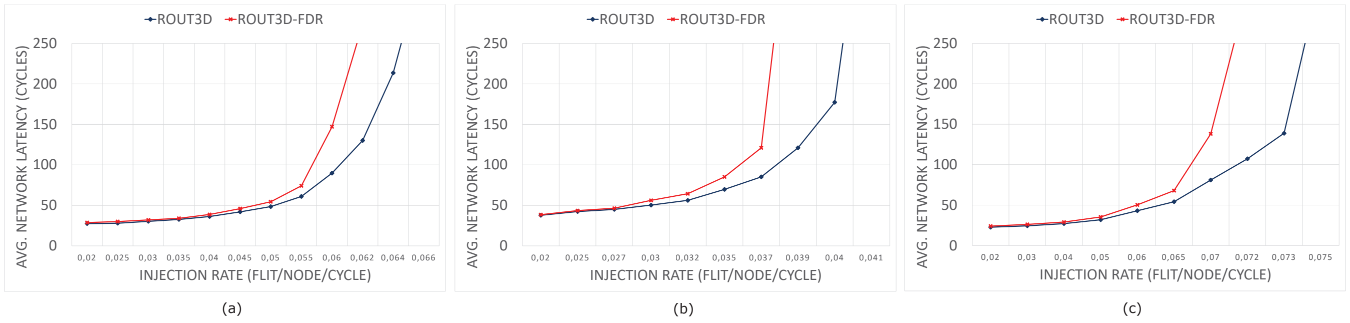

Fig. 5. Latency of ROUT3D-DRF and baseline ROUT3D under (a) uniform traffic, (b) bit complement traffic and (c) shuffle traffic.

fault points and running different benchmarks traffic.

VI. Evaluation And Analysis

In order to analyze the latency, hardware cost, and reliability of the proposed resilient RCU, we have extended the cycle-accurate Netmaker library [20] to support 3D-NoC router architectures. Netmaker is a library of parameterizable and synthesizable NoC routers written in SystemVerilog. This library was used to implement the detection and correction mechanism described in this paper, as well as the 3D-NoC router baseline described in [16].

The executed fault injection campaigns were performed in a partially and vertically connected 3D-NoC composed by 64 routers distributed in a 4x4x4 mesh architecture. The 3D-NoC planes are connected by 4 TSV pillars (25% vertical connections). This 3D-NoC routers are based on 4-flit deep FIFOs, and a packet size fixed in 5 flits (FLITs is an acronym for FLow control unITs, the logical unit of information of NoC packets). The routing algorithm uses the same configuration of VCs presented in [16]. When several candidates are available, the routing algorithm selects the least congested output port, based on a local congestion metric. For each simulation, the 5-flit synthetic packets are injected and wait until 100000 of them are received under a pessimistic fault injection rate of 5%.

We have tested this 3D-NoC built with three different NoC routers. The first circuit is the baseline router here termed ROUT3D. The second circuit identified as ROUT3D-TMR (Triple Modular Redundancy), is a version of the ROUT3D with TMR-hardened RCU in each port of the NoC routers. Finally, the third circuit implements the architecture described in Section IV-A. We refer to this circuit as ROUT3D-FDR (Fault detection circuit, Double sampling, and Rerouting).

The gates at which faults will be injected, the clock cycle and the duration of each fault are randomly chosen by the testbench. The probability of erroneous routing occurring at each route computation is fixed to a certain value. Faults are injected in the three stages of the RCU performing single, double and triple faults. Transient faults might occur at the same clock cycle in different routers, but in this experiment has not been considered that different faults can hit two RCUs from the same router.

A. *Latency Results*

The performance of the proposed RCU architecture is estimated through the average packet latency under three different traffic patterns (Uniform, Bit-complement, Shuffle). The resulting performance comparison is shown in Figure 5 when there are faults in the ROUT3D-FDR and no faults in the ROUT3D baseline. The degradation in performance is noticed because to recover from misrouting the VC, the same input port needs to be blocked to avoid the reception of new headers while executing the rerouting routine. The tendency is that packet's delay increases with the injection rate. This can be explained by the fact that a higher injection rate means a higher probability that new packets enter the router and use the routing unit instead of the rerouting mechanism.

B. *Hardware Synthesis Results*

In order to evaluate the hardware overhead of the proposed resilient RCU solution, we have performed two syntheses. In the first synthesis, we estimated the area and power overhead when all designs are setup to work with an operating frequency of 1GHz, a power supply of 1V, and a commercial ST FD-SOI $28nm$ library. In the second synthesis, the tool is configured to achieve the maximum operating frequency based on the critical path delay. The results for both syntheses are summarized in Table I. The three types of routers previously described were considered: 5-port 2D routers, 6-port 3D routers with one vertical connection, and 7-port 3D routers with both Up and Down vertical connections.

The area and power overhead showed in the Table I puts in evidence the low hardware overhead of the proposed method. We can see that for the 5, 6 and 7-port routers from ROUT3D-FDR, the area overhead increases around 4% while the ROUT3D-TMR increase by 12%. It is important to note that while the ROUT3D-FDR needs only an additional sample (double-sampling) to register the results of RCU and an additional circuit FDC to detect fault, the ROUT3D-TMR needs to triplicate all logic from RCU per input port as well as an additional voter.

In order to determine the maximum operating frequency that the 3D-NoC routers can achieve, we perform a sequence of syntheses increasing the operating frequency without time violation. The maximum operating frequency is shown in

Table I
HARDWARE SYNTHESIS RESULTS

Size	**ROUT3D**			**ROUT3D-TMR**			**ROUT3D-FDR**		
(# Ports)	Area (μm^2)	Power (mW)	Max. Freq (MHz)	Area (μm^2)	Power (mW)	Max. Freq (MHz)	Area (μm^2)	Power (mW)	Max. Freq (MHz)
5-Port	14945	5.8192	2380	16802	6.5628	2380	15564	6.2175	2372
6-Port	19036	7.8411	2300	21309	8.5576	2296	19738	8.1612	2290
7-Port	23888	9.6362	2200	26613	10.4897	2200	24681	9.9963	2190

columns "***Max. Freq***" of Table I. In this Table, we can see that the ROUT3D-TMR maintains practically the same maximum operating frequency while the ROUT3D-FDR decreases around 0.4% when both are compared to the baseline router. This is because the ROUT3D-FDR needs to decide which sample will be selected and if the packet needs to be rerouted after the FDC's analysis.

VII. CONCLUSION

We have proposed a mechanism to detect and correct transient faults in the route computation unit before affected packets leave the faulty router. Misrouting is detected combining double-sampling technique with a virtual channel allocator optimization and an extra fault detection circuit. Instead of relying on packet retransmission, a simple and safe rerouting mechanism allows for error recovery. In order to minimize the implementation cost, the same routing logic is used to service all the rerouting requests of one input port.

To validate the proposed technique, we also introduced a method that mimics the effects of transient faults in the NoC's gate-level netlist. Results obtained from a partially and vertically connected 3D-NoC showed that all faults can be corrected locally at the expense of a minimal latency increment. Indeed, rerouting is triggered by an efficient detection approach. Moreover, the measured area and power overhead including double-sampling, rerouting, and fault detection circuit was 4.1% and 6.8% respectively. These results suggest the proposed method is an appealing alternative to traditional TMR-based approaches. Future work includes protecting other critical modules from the NoC router control path such as the virtual channel allocator and switch allocator, offering a comprehensive solution to deal with transient faults.

REFERENCES

[1] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference*, 2001, pp. 684–689.

[2] A. B. Ahmed and A. B. Abdallah, "Graceful deadlock-free fault-tolerant routing algorithm for 3d network-on-chip architectures," *Journal of Parallel and Distributed Computing*, vol. 74, no. 4, pp. 2229 – 2240, 2014.

[3] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and L. S. Peh, "Research challenges for on-chip interconnection networks," *IEEE Micro*, vol. 27, no. 5, pp. 96–108, Sept 2007.

[4] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proceedings International Conference on Dependable Systems and Networks*, 2002, pp. 389–398.

[5] J. Howard, S. Dighe, S. R. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. K. De, and R. V. D. Wijngaart, "A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 173–183, Jan 2011.

[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, Jan 2011.

[7] 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 Twelfth International Symposium on High-Performance Computer Architecture, 2006.*, Feb 2006, pp. 5–16.

[8] X. Zhang, M. Ebrahimi, L. Huang, and G. Li, "Fault-resilient routing unit in nocs," in *2015 28th IEEE International System-on-Chip Conference (SOCC)*, Sept 2015, pp. 164–169.

[9] K. N. Dang, M. Meyer, Y. Okuyama, A. B. Abdallah, and X.-T. Tran, "Soft-error resilient 3d network-on-chip router," in *2015 IEEE 7th International Conference on Awareness Science and Technology (iCAST)*, Sept 2015, pp. 84–90.

[10] X. Zhang, M. Ebrahimi, L. Huang, G. Li, and A. Jantsch, "A routing-level solution for fault detection, masking, and tolerance in nocs," in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, March 2015, pp. 365–369.

[11] L. Huang, X. Zhang, M. Ebrahimi, and G. Li, "Tolerating transient illegal turn faults in nocs," *Microprocess. Microsyst.*, vol. 43, no. C, pp. 104–115, Jun. 2016.

[12] C. Chen and S. D. Cotofana, "A low cost method to tolerate soft errors in the noc router control plane," in *2013 IEEE International SOC Conference*, Sept 2013, pp. 374–379.

[13] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das, "Exploring fault-tolerant network-on-chip architectures," in *International Conference on Dependable Systems and Networks (DSN'06)*, June 2006, pp. 93–104.

[14] R. Parikh and V. Bertacco, "Formally enhanced runtime verification to ensure noc functional correctness," in *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2011, pp. 410–419.

[15] A. Charif, N. E. Zergainoh, and M. Nicolaidis, "Addressing transient routing errors in fault-tolerant networks-on-chips," in *2016 21th IEEE European Test Symposium (ETS)*, May 2016, pp. 1–6.

[16] A. Charif, N.-E. Zergainoh, A. Coelho, and M. Nicolaidis, "Rout3d: A lightweight adaptive routing algorithm for tolerating faulty vertical links in 3d-nocs," in *22th IEEE European Test Symposium (ETS'17)*. ACM IEEE, 2017, pp. 1–6.

[17] M. Bahmani, A. Sheibanyrad, F. Pétrot, F. Dubois, and P. Durante, "A 3D-NoC router implementation exploiting vertically-partially-connected topologies," *Proceedings - 2012 IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2012*, pp. 9–14, 2012.

[18] F. Dubois, A. Sheibanyrad, F. Petrot, and M. Bahmani, "Elevator-First: A Deadlock-Free Distributed Routing Algorithm for Vertically Partially Connected 3D-NoCs," *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 609–615, mar 2013.

[19] M. Nicolaidis, "Double-sampling design paradigm - a compendium of architectures," *IEEE Transactions on Device and Materials Reliability*, vol. 15, no. 1, pp. 10–23, March 2015.

[20] R. Mullins. (2009) Netmaker. [Online]. Available: http://www-dyn.cl.cam.ac.uk/~rdm34/wiki