Reliability Assessment and Quantitative Evaluation of Soft-Error Resilient 3D Network-on-Chip Systems

Khanh N. Dang, Michael Meyer, Yuichi Okuyama, and Abderazek Ben Abdallah
The University of Aizu
Graduate School of Computer Science and Engineering
Aizu-Wakamatsu 965-8580, Japan
Email: {d8162103, d8161104, okuyama, benab}@u-aizu.ac.jp

***Abstract*—Three-Dimensional Networks-on-Chips (3D-NoCs) have been proposed as an auspicious solution, merging the high parallelism of the Network-on-Chip (NoC) paradigm with the high-performance and low-power cost of 3D-ICs. However, as technology scales down, the reliability issues are becoming more crucial, especially for complex 3D-NoC which provides the communication requirements of multi and many-core systems-on-chip. Reliability assessment is prominent for early stages of the manufacturing process to prevent costly redesigns of a target system. In this paper, we present an accurate reliability assessment and quantitative evaluation of a soft-error resilient 3D-NoC based on a soft-error resilient mechanism. The system can recover from transient errors occurring in different pipeline stages of the router. Based on this analysis, the effects of failures in the network's principal components are determined.**

***Keywords*-Reliability Assessment; Fault-tolerant; 3D Network-on-Chip; Soft-Error; Architecture.**

I. Introduction

Global interconnects are becoming the largest performance bottleneck for high-performance Multi/Many-core Systems-on-Chip (MSoCs). For more than a decade, Network-on-Chip (NoC) interconnects have been proposed as a promising solution for future MSoC designs [1]. The NoC paradigm offers more scalability than conventional shared-bus interconnects and allows more processing elements (PEs) to be efficiently integrated into a single chip. Despite the higher scalability and parallelism offered by a NoC system over traditional shared-bus based systems, it is still not an ideal solution for future large scale MSoCs. This is due to some limitations such as high power consumption and low throughput from NoCs with large dimensions. Taking NoCs to the third dimension has been proposed to deal with the above problems, as it was a solution offering lower power consumption and higher speeds [2], [3].

As feature sizes and supply voltages continually decrease, systems that have implemented these interconnects have become more vulnerable to soft errors. *Shivakumar et al.* [4] analyzed the transient error trends for smaller transistors and showed that the occurrence rate of transient faults is significantly higher than permanent faults. In particular, they expect the transient error rate for combinational logic to increase dramatically. Currently, soft error handling is mostly focused on memory and latches error which can be tackled by Error Correction Code. Therefore, future integration systems need a suitable solution to deal with soft errors.

Recently, numerous techniques have been proposed to handle soft errors [5]–[8] in NoCs; however, the reliability of the proposed techniques is still not well investigated. The most conventional method to evaluate the error resilience is injecting faults into the system and checking the output's accuracy. This kind of evaluation, especially at the gate level, requires a massive amount of time and computing resources. Moreover, the correction module is not sufficiently investigated in the reliability evaluation stage. Although the correction module helps to handle fault occurrences, it also suffers from an increase in fault probability.

In this paper, we present an efficient soft-error resilient mechanism and architecture for reliable 3D-NoC systems. We also present an accurate formulation of the reliability and vulnerability of the proposed 3D-NoC architecture against soft-errors. Our assessment method is based on the $Markov\ state\ model$ [9]. The fault-tolerance architecture is modeled as several states that include all of the possible failure cases. To switch between states, the $repairability$, and the failure rate is required. Based on the state model, a reliability function can be generated using probability functions. Moreover, this paper presents a metric named the Reliability Acceleration Factor (RAF). This parameter is used to separate the reliability from the technology parameters, or operating conditions. By using the characteristics of the architecture and its fault-tolerance mechanism, the RAF value only represents the efficiency of the fault-tolerance mechanism alone.

II. System Architecture Overview

The 3D-Network-on-Chip (3D-ONoC) router block diagram is shown in Fig. 1. The router has three pipeline stages: (1) BW (Buffer Writing), (2) NPC/SA (Next Port Computation and Switch Allocation), and (3) CT (Crossbar Traversal).

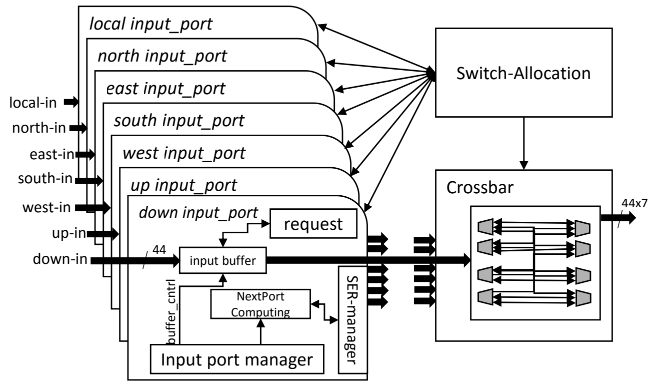


Figure 1: Proposed Soft Error Resilient 3D-NoC router architecture.

The router architecture contains seven *Input-port* modules for each direction in addition to the *Switch-Allocation* and the *Crossbar* module, which handle the transfer of flits to the next node. The *Input-port* module is composed of two main elements: *Input-buffer* and the *Next-Port-Routing* module. Incoming flits from the previous node or the local core, are first stored in the *Input-buffer*. Since 3D-ONoC uses a *Look-Ahead* routing algorithm, the output port value is already calculated by the previous node/core. Therefore, the *output-port* value is sent with request signal to the *Switch Allocation* module. At the same time, the *Next-Port-Computing* module uses the routing information (*destination* and *output-port*) to calculate the routing i nformation for the next node, labeled as *next-output-port*. With the grant from *Switch Allocation*, the flit will be sent through the *Crossbar* to the next node together with the *next-output-port* value.

A. Soft-error Resilient Mechanism

Our main goal in proposing the SER-3DR (Soft-Error Resilient 3D-Network-on-Chip Router) is to develop a highly-reliable and low-cost technique to recover from soft-errors in all pipeline stages of the router.

For the errors in the pipeline stages, we propose a Soft Error Resilient Algorithm (SERA) , as shown in Algorithm 1. In the baseline OASIS-NoC, the router has three pipeline stages: BW (Buffer Writing), NPC/SA (Next Port Computing and Switch Allocation in parallel) and CT (Crossbar Traversal). Since the NPC/SA stage (*Routing and Arbitrating*) is where a majority of the complex combinational logic exists and it executes routing for the router, this stage is the one we selected for the SERA algorithm.

As illustrated in Algorithm 1, SERA routes a flit from an input port to an output port. The input flit's data (in_flit) is first written into the input buffer during the BW (Buffer Writing) stage (line 1). Second, SERA executes the first-time NPC (NextPortComputing) and SA (SwitchAllocation) stages which output the next_port[1] and grants[1] respectively (lines 3-4). The NextPortComputing computes the routing path for the next node, similar to the look-ahead routing algorithm, and the SwitchAllocation handles the input port to output port routing. Third, the redundant processes of the NPC and SA are performed with these outputs: next_port[2] and grants[2] (lines 4-5). In the next step, SERA compares the outputs of the original and the redundant processes. If next_port[1] is different from next_port[2], a soft-error occurred in the NPC, the algorithm then calculates the NPC a third time and uses majority voting to decide the final value (line 7-8). Otherwise, the final value is assigned as the first result (line 10). The SA is also processed in a similar fashion to the NPC: it starts by determining the occurrence of any errors, then voting on a value or assigning the first value (line 12-17). After detection and recovery, SERA finishes with crossbar traversal (line 18). The flit will be forwarded to the next node in the routing path or to the local core.

Algorithm 1: Algorithm for SER-3DR

```
   // input flit's data
   Input: in_flit
   // output flit's data
   Output: out_flit
   // Write flit's data into buffers
1  BufferWriting(in_flit)
   // Compute first time of NPC and SA
2  next_port[1] = NextPortComputing(in_flit)
3  grants[1] = SwitchAllocation(in_flit)
   // Compute redundant of NPC and SA
4  next_port[2] = NextPortComputing(in_flit)
5  grants[2] = SwitchAllocation(in_flit)
   // Compare orginal and redundant to detect soft-error
   // Soft-error on NPC
6  if (next_port[1] $\neq$ next_port[2]) then
       // roll-back and recalculate NPC
7      next_port[3] = NextPortComputing(in_flit)
8      final_next_port = MajorityVoting(next_port[1,2,3]);
9  else
       // No soft-error on NPC
10     final_next_port = next_port[1]
11 end
   // Soft-error on SA
12 if (grants[1] $\neq$ grants[2]) then
       // roll-back and recalculate SA
13     grants[3] = SwitchAllocation(in_flit)
14     final_grants = MajorityVoting(grants[1,2,3])
15 else
       // No soft-error on SA
16     final_grants = grants[1]
17 end
   // After detection and recovery, the algorithm
      finishes with CT
18 out_flit = CrossbarTraversal(in_flit, final_next_port, final_grants);
```

III. Reliability Assessment

A. Mean Time Between Failure

In this section, we present a methodology to calculate the reliability of the system using the Markov State Model and the reliability function [9]. The reliability of a system with respect to soft errors can be evaluated using the Mean-Time-Between-Failure calculation that follows:

$$\mathbf{MTBF} = \lim_{s \to 0} R(s) \qquad (1)$$

where $R(s)$ is the reliability function of the system in the Laplace domain [9], which can be calculated based on the Markov state models as shown in Fig. 2. Each state S_i of the Markov model represents a possible status of the system. The status is defined as each case where an element of the system fails, which can lead to unpredictable operations. If the operation in S_i state is correct, we define this state as “healthy”. If the system is unable to operate correctly in state S_i, we define this state as “faulty”.

Assuming the system has a set $\mathbb{S}$ which includes states from S_0 to S_m in its Markov state model. We use S_0 to represent the initial state of the system. The set of healthy states and set of faulty states are defined as:

$$\mathbb{H} \triangleq \{S_i \in \mathbb{S} | \text{the system works correctly}\} \tag{2}$$

$$\mathbb{F} \triangleq \{S_i \in \mathbb{S} | \text{the system not working}\} \tag{3}$$

where:

- $\mathbb{H}$ is the set of states which still maintain the operation of the system,
- $\mathbb{F}$ is the set of states which the system fails.

The reliability function $R(s)$ in Equation 1 is defined as follows:

$$R(s) = P(\mathbb{H}) = 1 - P(\mathbb{F}) \tag{4}$$

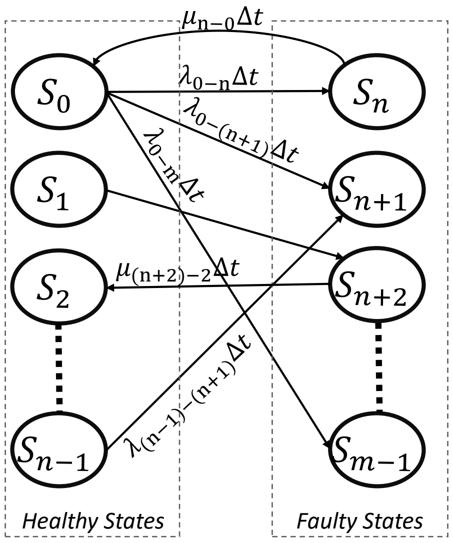


Figure 2: A Markov-state reliability model for a m states system with n non-failure states.

As shown in Fig. 2, each state S_i represents a status of the system during operation. There are two subsets, healthy states $H = \{S_0, S_1, ..., S_{n-1}\}$ and faulty states $F = \{S_n, S_{n+1}, ..., S_{m-1}\}$. The transition rate are defined as follows:

- λ is the fault rate a component in the system which can represent for a transition from a element of set $\mathbb{H}$ to a element of set $\mathbb{F}$.
- μ is the repair rate a component in the system which can represent for a transition from a element of set $\mathbb{F}$ to the element of set $\mathbb{H}$.

The final MTBF can be calculated based on the healthy states as:

$$\mathbf{MTBF} = \lim_{s \to 0}(P(\mathbb{H}(s))) = \lim_{s \to 0} \sum_{i=0}^{n-1} S_i(s) \tag{5}$$

B. Fault Model

Before analyzing the reliability of the system and calculating the MTBF, we need make two assumptions about the failure of the system.

- The system starts with a default all components healthy state. In Fig. 2, the initial status is: $S_0 = 1$ and $S_i = 0$ with $i \neq 0$.
- The transition rate λ and μ are constants and perdetermined.

However, the fault rate and repair rate depend on the technology parameters, running environment and operating circumstances. In order to separate them from the fault rate value, we propose these assumptions:

- The fault rate has a linear relationship to the area cost of the module.
- The fault rate has a linear relationship to the operating time of the module.
- The fault rate is affected after the module is attached to a system.

In this fashion, the fault rate only depends on the area cost, the operating time and the efficiency of the reduction of the fault rate that is provided by the fault-tolerance mechanism. Thus, for a system with k components, its fault rate is given by:

$$\lambda_{system} = \sum_{i=1}^{k} f_i \mathbf{OR}_i \mathbf{AR}_i \lambda_{unit} \tag{6}$$

where:

- $unit$ is a select module as a reference for calculation. In our method, we use the original design as a $unit$.
- $\mathbf{OR}_i$ is the operating time ratio of component i to $unit$.
- $\mathbf{AR}_i$ is the area cost ratio of component i to $unit$.
- f_i is the changing rate caused by attaching the module i to the system. If the system has no impact to the failure of module i or module i is standalone, $f_i = 1$. If the system has a fault-tolerance mechanism for module i, $f_i < 1$.

Soft errors occur over a short period of time, typically within a single clock cycle. Because of this, the combinational logic has the ability to self-correct soft errors. On the other hand, the memory units (latch, flip-flop,...) still record failures.

C. Reliability Evaluation

In order to evaluate a fault-tolerance method, we introduce a parameter named the Reliability Acceleration Factor (RAF), which is defined as:

$$\mathsf{RAF} = \frac{\mathbf{MTBF}_{FT}}{\mathbf{MTBF}_{original}} \tag{7}$$

where $\mathbf{MTBF}_{FT}$ and $\mathbf{MTBF}_{original}$ are *Mean Time Between Failures* of the Fault-Tolerant and the original system, respectively.

Based on analysis of the operation of the original and fault-tolerant system, we can obtain both MTBFs using Equation 5. Moreover, by using Equation 7, the RAF parameter is separate from the fault rate of the *unit* module. In this fashion, the RAF parameter only represents the efficiency of the fault-tolerance mechanism.

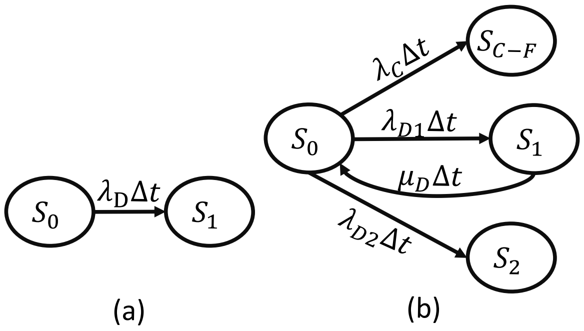


Figure 3: A simplified Markov-state reliability model for: (a) the original system; (b) the fault-tolerant (FT) system.

In order to evaluate a fault-tolerant system which consists of both original modules and correction modules, we model the fault-tolerant system to have a Markov state similar to Fig. 3 (b) with:

- S_0 is the initial state.
- S_1 in Fig. 3 (b) is the failure state of the original system which is given by a transition rate λ_{D1}.
- S_2 in Fig. 3 (b) is the failure state of the original system which is given by a transition rate λ_{D2}.
- S_{C-F} in Fig. 3 (b) is the failure state of the repair module.

Because of the protection from the fault-tolerance technique, the FT system can handle some kinds of faults. Therefore, we define the transition rates as:

- λ_D is the fault rate of the original system (D).
- λ_C is the fault rate of the repair module of the FT system.
- μ_D is the repair rate which is provided by the repair module (C).
- λ_{D1} is the part of the fault rate in the original which is handled by the repair module (C).
- λ_{D2} is the part of the fault rate in the original which the repair module (C) cannot correct.

Based on the Markov state of the two systems, as shown in Fig. 3, the MTBFs are given as:

$$\mathbf{MTBF}_{original} = \frac{1}{\lambda_D} \tag{8}$$

and

$$\mathbf{MTBF}_{FT} = \frac{1}{\lambda_{D2} + \lambda_C} \tag{9}$$

By applying Equations 6, 8, and 9 to Equation 7, we have:

$$\mathsf{RAF} = \frac{1}{(f_D \times \mathbf{OR}_D \times \mathbf{AR}_D) + (\mathbf{OR}_C \times \mathbf{AR}_C)} \tag{10}$$

where:

- $f_D = \lambda_{D2}/\lambda_D$ is the ratio of the failure rate of the original system after and before applying the fault-tolerance mechanism.
- $\mathbf{OC}_x$ is the ratio of operation of the module x to the original system (D).
- $\mathbf{AR}_x$ is the ratio of area cost of the module x to the original system (D).

Equation 10 shows the RAF function based on the architecture modifications in area cost and operation and the reduction of the failure rate.

IV. Design and Evaluation Results

Our proposed system (SER-3DR) is integrated into the 3D-ONoC [2], [3]. We designed the system in Verilog-HDL, and synthesized it using the 45nm technology library. We evaluated the hardware complexity, power consumption and speed. We also evaluated performance using three benchmarks: Matrix-multiplication, Transpose, and Uniform were selected. For comparison, we also implemented and simulated the baseline LAFT-OASIS [2], and Triple Modular Redundancy of the NPC/SA based on OASIS (TMR) [5]. In order to demonstrate the reliability, we analyze our architecture to provide the RAF parameter in Eq. (10) and provide some comparison with another mechanism.

A. Hardware Complexity

Table I depicts the implementation result of a single router of the original OASIS system, the TMR router [5], and the proposed SER-3DR router. The router is designed for a 3D Mesh with 7 ports, 32 bit flit-width, Stall-Go flow control and wormhole forward mechanism. When compared with the original LAFT-OASIS router architecture, the SER-3DR requires slightly more logic area cost (14.98%). The TMR router costs 45.20% more because it triplicates the NPC and SA stage. The frequency decreases from 801.28 MHz to 655.74 MHz (-18.16%). Also, the TMR system increases the power consumption to 30.31 mW (+18.30%). The proposed design slightly increases the power consumption from the baseline's 25.62 mW to 27.13 mW (+5.90%).

Table I: Hardware complexity comparison results.

Design	Max Freq. (MHz)	Total Power mW)	Logic's area (μm^2)	# TSVs
LAFT OASIS	801.28	25.62	14,920	164
TMR-OASIS	763.36	30.31	21,664	164
SER-3DR	655.74	27.13	17,154	164

B. Network Performance Evaluation

For this evaluation, we used the three benchmarks over five injection rates : 0%, 8.33%, 16.67%, 11.11%&6.67% (in Routing and Switch Allocator) and 33%. The evaluation results with Transpose and Uniform are shown in Fig. 4 (a and b), respectively. We perform these benchmarks for 3 models (SER-3DR, LAFT-OASIS and the TMR). The network transmission time or average delay is presented as a bar graph. We also inject the soft-errors inside the baseline model (LAFT-OASIS) and measure the transmission time. Its *failure time* and complete *finish time* are depicted in line graph format.

For the Transpose benchmark in Fig. 4(a), we found that the average latency slightly increased from 20.113 $cycless$ to 20.505 $cycles$ (+1.95%) for an error injection rate of 0%. With different error injection rates, we can see that the average latency slightly increased from 20.505 $cycles$ for a 0% error rate to 21.092 $cycles$ for a 33% error rate. Uniform benchmark has a 9.06% increase in transmission time with an absence of faults, while Matrix has a 10.02% additional transmission time. In the faulty cases, SER-3DR requires additional intervals for detecting and recovery.

To perform the throughput evaluation, we also used the above benchmarks with five injection rates as shown in Fig. 4 (c and d). For the Uniform benchmark, the throughput is slightly degraded due to the short packet length which increases the impact of redundant cycles.

C. Reliability Comparison

In this section, we compare the soft error resilience technique designed for the two modules: Routing and Switch Allocation. We select the TMR and two architectures from [7], [8]. The architecture by Prodromou et al. [8] and Yu et al. [7] use monitors to verify the accuracy of the output values. To verify the output value, they predefine a set of rules based on the architecture of the monitored modules. If the output value violates one of the rules, the monitor module can determine the failure and decide the next step. The architecture by Yu et al. [7] allows the system to re-execute the task again to correct the failure. Prodromou et al. [8] does not provide any recovery method for the detected failure. Although the set of faults for the routing and arbitrating modules consists of 7 and 13 faults, we assume both architectures can handle 100% of transient faults in this evaluation.

For our proposal [5] and Yu et al. [7], we use Eq. 10 to calculate the RAF value. The reliability function of TMR is obtained from [9]. The original module is not modified to implement the fault-tolerance architecture. Therefore, the area cost and operating ratios (AR_D and OR_D) of the protected module both equal 1. Since the operation of the correction module follows the operation of the protected module, the operating ratios of the correction module OR_C are both equal to 1. The area ratios AR_C of the correction module are shown in Table II. Since our proposal can handle 100% kind of the error, we assume that Yu et al. [7] also has the same capacity.

As shown in Table II, our proposed architecture has a medium area cost overhead (54.46%). Both monitor based architectures have a small area overhead (9% and 3%). The TMR architecture has the largest area cost (+204.33%) but it has the least impact on performance and architecture modification. Although monitor based architectures provide a small area overhead, they only cover the faults which violate their set of rules. Therefore, a new architecture requires another set of rules to handle soft errors. On the other hand, our proposal and TMR can cover 100% kind of transient fault and adapt to any architecture without significant modifications.

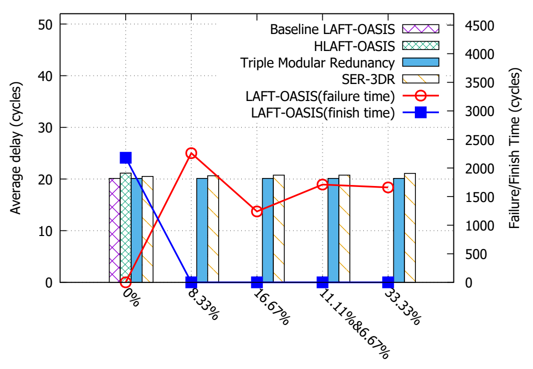
In terms of RAF (Reliability Acceleration Factor), the TMR attains a value of around 1.33. Our architecture provides a RAF of 1.84. For monitor-based architectures, Yu et al. [7] provides the best protection with 11.11; however, this value is attained under the assumption that the design will cover 100% of failures. The architecture by Prodromou et al. [8] does not improve due to the fact that they did not implement a recovery module.

V. Conclusion

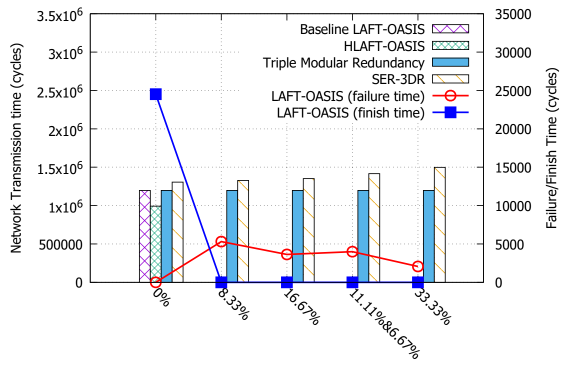
In this paper, we presented an accurate reliability assessment and quantitative evaluation of a soft-error resilient 3D-NoC system based on a soft-error resilient mechanism. The system can recover from transient errors occurring in different pipeline stages of the router. Evaluation results show that the system has about 1.84 times improvement in MTBF while handling 100% of all tested error types. Moreover, the system can achieve a high level of transient error protection with a small latency increase of 18.16% when compared to the baseline non-protected system.

References

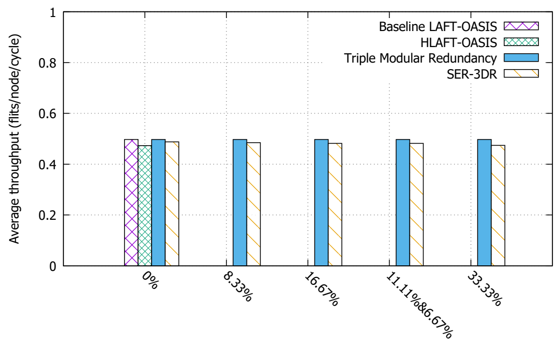
[1] A. B. Abdallah and M. Sowa, "Basic Network-on-Chip Interconnection for Future Gigascale MCSoCs Applications: Communication and Computation Orthogonalization," in *Proc. of the Symposium on Science, Society, and Technology (TJASSST2006)*, pp. 1–7, 2006.



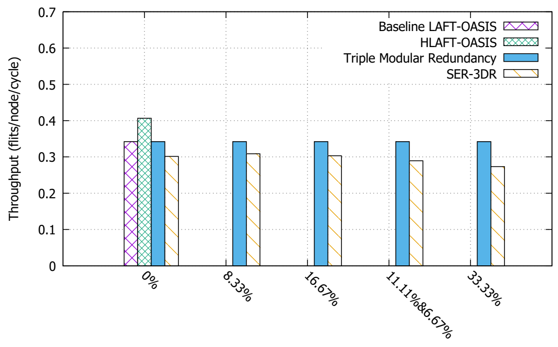
(a) Average Delay of Transpose.



(b) Network Transmission Time of Uniform.



(c) Throughput of Transpose.



(d) Throughput of Uniform.

Figure 4: Simulation's Results.

Table II: Comparison of soft-error resilient mechanisms for routing and arbitrating modules in Network-on-Chip router.

Model	TMR for OASIS [5]	Yu et al. [7]	Prodromou et al. [8]	Our proposal [5]
Mechanism	Majority Voting	Monitor	Monitor	Monitor
Area Overhead	204.33%	9%	3% (average)	54.46%
$\mathbf{AR}_C$	0.0433	0.09	0.03	0.5446
RAF	$\simeq$ 1.33	$\simeq$ 11.11	$\simeq$ 1 (only detection)	1.84
Delay	+0	+0 cycle (no fault) +1 cycle (recovery)	0% (only detection)	+1 cycle (redudancy) +2 cycle (recovery)
Fault Coverage	100% of permanent and transient	design specific (7 faults)	design specific (13 faults)	100% transient
Reovery method	immediately	re-execution	unsupport	re-execution

[2] A. B. Ahmed and A. B. 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, no. 3, pp. 1507–1532, 2013.

[3] 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, no. 4, pp. 2229–2240, 2014.

[4] P. Sivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on soft error rate of combinatorial logic," in *Proc. Intl. Conf. Dependable Sys. & Networks DSN02*, pp. 23–26, 2002.

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

[6] K. N. Dang, Y. Okuyama, and A. B. Abdallah1, "Soft-error resilient network-on-chip for safety-critical applications," in *2016 International Conference on IC Design and Technology (ICICDT)*, pp. 1–4, June 2016.

[7] Q. Yu, M. Zhang, and P. Ampadu, "Addressing network-on-chip router transient errors with inherent information redundancy," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 4, p. 105, 2013.

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

[9] M. L. Shooman, *Reliability of computer systems and networks: fault tolerance, analysis, and design*. John Wiley & Sons, 2003.