A Comprehensive Reliability Assessment of Fault-Resilient Network-on-Chip Using Analytical Model

Khanh N. Dang, Akram Ben Ahmed, Xuan-Tu Tran, Yuichi Okuyama, and Abderazek Ben Abdallah

***Abstract*— The component's failure in network-on-chips (NoCs) has been a critical factor on the system's reliability. In order to alleviate the impact of faults, fault tolerance has been investigated in the recent years to enhance NoC's robustness. Due to the vast selection of fault-tolerance mechanisms and critical design constraints, selecting and configuring an appropriate mechanism to satisfy the fault-tolerance requirements constitute new challenges for designers. Consequently, reliability assessment has become prominent for the early stages of manufacturing process to solve these problems. This paper approaches the fault-tolerance analysis by providing an analytical model to approximate the lifetime reliability and compares it with a system-level simulation. Based on the proposed approach, we measure the fault-tolerance efficiency using a new parameter, named reliability acceleration factor. The goal of this paper is to provide an efficient and accurate reliability assessment to help designers easily understand and evaluate the advantages and drawbacks of their potential fault-tolerance methods.**

***Index Terms*— Analytical model, architecture and design, fault tolerance, reliability analysis.**

I. Introduction

IN THE past few years, the benefits of network-on-chips (NoCs) [1] have been demonstrated as a prominent paradigm for future IC designs. While the NoC paradigm has been increasing in popularity with several commercial chips, it is threatened by the decreasing reliability of aggressively scaled transistors. Gate widths are nearing the molecular scale, resulting in breakdown and wear out in end products [2], [3]. Moreover, the smaller anticipated fabrication geometry, the low supply voltage, and the rising of power density [4] cause the architecture to become more vulnerable and more sensitive to faults. As a result, future NoC systems may require sufficient, and sometimes complex, fault-tolerant (FT) mechanisms to ensure the system's reliability.

Although there is a substantial number of fault-tolerance works for NoCs, which were summarized in [5], their reliability efficiencies still need to be carefully investigated before selecting the appropriate method. Moreover, several works have different configurations (e.g., number and level of modular redundancies [6] and checkpoint/recovery interval [7]) and design tradeoffs (additional power consumption, performance degradation, and extra area cost), which create difficulties of selecting the most suitable FT technique for a given system. To reduce the risk of redesign when the final system does not match the initial requirements, these fault-tolerance methods need to be sufficiently evaluated. Notably, a well-recognized gap between the growth of system complexity (58%/year) and design productivity (21%/year), which is widened every year [8], demands designers to carefully select the potential FT method before implementation. This aims to avoid any potential waste of time and resources. To ensure the ability of the fault-tolerance method, most works are verified by injecting faults and checking the correctness under the created faulty situations [5], [9]. This verification shows the capacity of the fault-tolerance scheme; but, it does not provide exact efficiency results (e.g., the improvement of lifetime, the potential of the correcting module being defected, or the impacts when attaching the FT scheme into the complete system). This is because when designers want to implement a fault-tolerance scheme onto the system, they need to know the enhancement capabilities and the tradeoff of the given scheme [10], [11]. Moreover, by introducing new materials and scaling down the feature sizes [4], the reliability challenges may also change. In the 2015 ITRS Report [4], the long-term prediction (2023–2030) is to shift to unreliable devices-based system, which demands in-depth studies and analyses to satisfy the reliability requirements.

Design with reliability awareness [11] is a methodology to ensure the robustness of the system, which needs to be early predicted and evaluated carefully. According to [10], there are five basic phases in reliability assessment: system definition, preliminary design, detailed design, fabrication, assembly, integration, and test (FAIT), and product/support phase. Notably, reliability prediction in the three early phases is important to prevent the wasted time of manufacturing and designing (FAIT phase).

For the early reliability assessment, there are three basic methods: physical analysis, system-level simulation, and

Manuscript received February 1, 2017; revised May 2, 2017 and June 24, 2017; accepted July 30, 2017. This work was supported by the VLSI Design and Education Center, The University of Tokyo, Japan, in collaboration with Synopsis, Inc., and Cadence Design Systems, Inc., and in part by the Competitive Research Funding, The University of Aizu, under Grant P-11-2016 and Grant P-2-2017. The work of X.-T. Tran was supported by Nafosted under Project 102.01-2013.17. *(Khanh N. Dang and Abderazek Ben Abdallah contributed equally to this work.) (Corresponding author: Khanh N. Dang.)*

K. N. Dang, Y. Okuyama, and A. Ben Abdallah are with the Adaptive Systems Laboratory, Graduate School of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu 965-8580, Japan (e-mail: d8162103@u-aizu.ac.jp; benab@u-aizu.ac.jp).

A. Ben Ahmed is with the Department of Information and Computer Science, Keio University, Yokohama 223-8522, Japan.

X.-T. Tran is with the Key Laboratory for Smart Integrated Systems, VNU University of Engineering and Technology, Hanoi 123106, Vietnam.

Digital Object Identifier 10.1109/TVLSI.2017.2736004

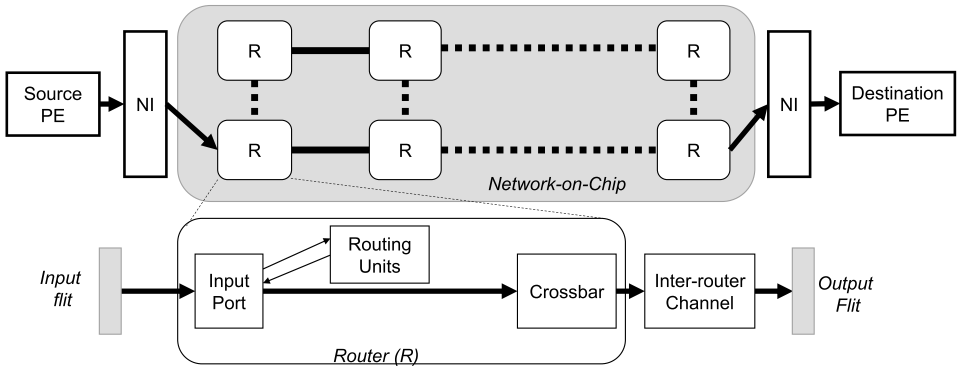

Fig. 1. NoC simplified block diagram.

analytical model. The physical-analysis method [12], [13] focuses on the device reliability under specific conditions. The whole system's reliability can be synthesized from its components' reliability. A major drawback of physical analysis is the complexity of evaluation for large systems, which may involve a huge amount of analyses. System-level simulation [10], [14] requires having fully implemented system characteristics, which is difficult to be obtained in the system definition or the preliminary design phase. On the other hand, the analytical model is popular in the computer network's reliability assessment [15], [16]; but, it is not widely applied in integrated systems. Specifically, NoC reliability assessment is still immature. Recently, the works in [17] and [18] have provided good analyses on through silicon via (TSV) failure and a baseline NoC system. Because of the continuous increase in terms of complexity, the NoCs assessment using the physical-analysis and system-level simulation is no longer suitable. Therefore, an efficient analytical method has become primordial.

In this paper, we present an analytical platform to evaluate FT NoC architectures. Instead of building a complex analysis, we aim to provide a simple and effective method to address the reliability of NoCs. The proposed method can work in conjunction with other reliability analyses, such as physical-level analysis, simulation evaluation, and system-level analysis. We build a strategy to help analyzing a complex network-based system in separated parts and merge them together. The range of applying this proposal can be from the design definition phase, through preliminary design and up to the detailed design phase. The main contributions of this paper are summarized as follows:

1) new method for analyzing the reliability of NoC systems using analytical model;
2) evaluations and analyses of the proposal for an NoC system;
3) verification using gate-level simulations.

The rest of this paper is organized as follows. Section II briefly presents the NoC concept and introduces the challenges in reliability and solutions. In Section III, we review the existing reliability assessment methodologies. In Section IV, we overview the Markov-state model in addition to the main assumptions and definitions needed for the proposed method. In Section V, we present our reliability assessment methodology. In Section VI, we provide the evaluation results. Section VII is dedicated to conclusions and future work.

II. NoC Architecture, Challenges, and Solutions

A. Network-on-Chip Architecture

In order to fully understand the proposed reliability assessment methodology, it is important to first grasp the main building blocks of a given NoC system. This aims to highlight the main differences between these blocks in order to understand later how the assessment methodology can be applied to each module.

Fig. 1 shows an NoC system with three main elements: 1) routers—they are connected to each other via point-to-point channels; 2) network interfaces (NIs)—constitute the interfaces between the routers and their attached processing elements (PEs); and 3) PEs—execute the program and they are connected to the network via NIs.

A router includes three main parts: input buffers, routing units, and a crossbar. Data are divided into a set of packets. Each packet consists of several flits, which are obtained by the flitizing process. A flit travels from a source to its destination through the network with the help of routers. Inside a router, an incoming flit is stored in an input buffer, routed by routing units (virtual channel, switch allocation, and intrarouter arbitrating) and physically forwarded to the next node by a crossbar channel. The flit is transmitted via an interrouter channel to the next router or to the attached NI. The routing path of a packet is decided by the routing unit. After completely transmitting a packet, the routing configurations of the packet are released for future packets.

B. Potential Faults and Effects on Network-on-Chip

Fig. 2 shows the potential challenges of NoCs. We divide them into two categories, soft errors and hard faults, due to their different characteristics and handling methods. The potential effects and common solutions are also depicted.

1) Soft Errors: This kind of error typically occurs over a short period of time. They are caused by crosstalk, radiation particles, cosmic rays, thermal neutrons, or noises [19], [20]. The potential effects of soft errors are data corruption and logic malfunction. Because of the incorrect arbitration or the damaged routing information, they may cause inaccurate routing

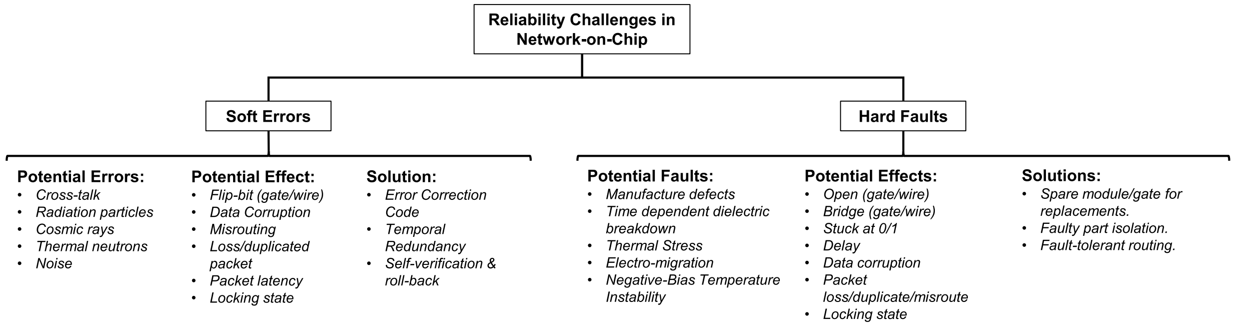

Fig. 2. Taxonomy of reliability challenges, potential effects, and solutions in NoC.

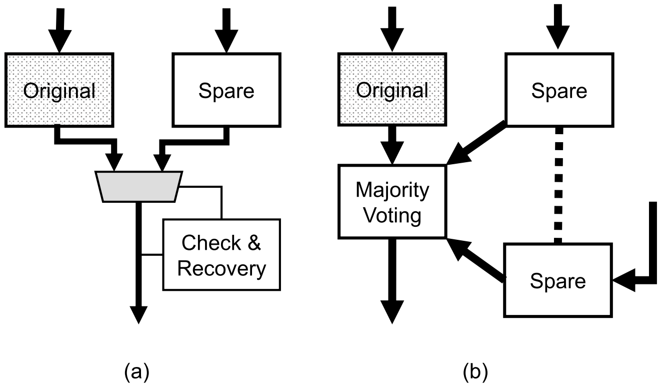

Fig. 3. Redundancy FT models. (a) Check and recovery. (b) Majority voting.

paths, dismiss, or duplicate flits. The inaccurate routing may even create a locked state, which crashes the NoC system.

2) Hard Faults: This kind of faults affects permanently the system and may occur during the manufacturing stage or operating lifetime [5], [20]. Typically, this kind of faults affects the operation of a gate/wire, which leads to an inaccurate output value. The causes can be electromigration, thermal stress, negative-bias temperature instability, time-dependent dielectric breakdown, or process variation. The most popular effects are open, bridge, and stuck at 1/0. NoCs also have their specific effects, such as data corruption, lost packets, duplicated packets, and miss routing. In some cases, hard faults can split the network, or create a locked state, which prevents the NoCs from working correctly.

C. Fault-Tolerant Technique

In [5], the fault-tolerance models and techniques for NoCs are surveyed and organized. Hereafter, we briefly summarize them and categorize them into two basic approaches.

1) *Redundancy:* It consists of temporal or spatial redundancy to handle the faults, as shown in Fig. 3.
2) *Self-Reconfiguration:* As represented in Fig. 4, the fault tolerance adapts to the occurrence of faults to alleviate their impact.

The *redundancy* technique can use a replicate of a given module as a backup when the original system fails, as shown in Fig. 3(a). It can also use multiple replicates run in parallel, which can be activated on the fly, as shown in Fig. 3(b).

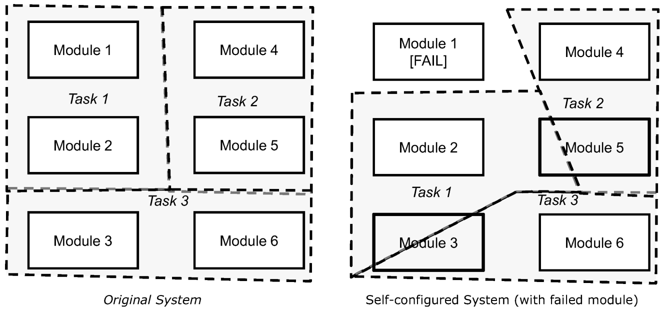

Fig. 4. Self-configuration FT models.

The self-configuration method reoptimizes the system to ensure the system's function. As represented in Fig. 4, if a module fails, its task can be migrated and shared with other healthy modules. Thus, the system can maintain its correct functionality.

Along with the recovery methods, one of the important criteria of fault tolerance is error detection. Depending on the reliability requirements, NoC systems can use an online [21], [22] or an offline [23], [24] error detection method.

III. Related Work

Table I shows the different methodologies used to analyze a given system's reliability. As previously mentioned, we categorized them into three basic methods: physical-level analysis, system-level simulation, and analytical models.

The physical-level analysis [12], [13], [28] can obtain accurate results for gates or small devices; however, it is time-consuming and requires huge computation resources to analyze complex systems. The system-level simulation can be performed by a Monte Carlo simulation [10] or a register transfer level-level simulation [5], [14], [29], [30]. In order to evaluate the system's reliability, the mean time to failure (MTTF) [11] estimation can be used. Beside analysis, design awareness is one of the important keys to obtain reliable systems [31]–[34].

The analytical models [3], [17], [35], [36] can be used to save a significant time wasted on redesign. The IEEE 1413.1 [10] recommends several methods: system reliability block diagrams, fault-tree Analysis, and Markov model for repairable systems. The most basic method is the system

TABLE I
RELIABILITY ASSESSMENT METHODOLOGIES

Method	Description	Type
Physical Failure Analysis [12]	The acceleration factors of the failure rate are defined as: Oxide, Metalization, Hot Carrier, Contamination, Package, EOS/ESD (Electrical Overstress/ Electrostatic Discharge) and Miscellaneous Failure Rate	physical-level
Soft Error Rate Estimation [13]	This simulation flow uses a random function to generate hit locations and perform analysis using HSPICE simulation. This aims to obtain the probability of soft errors with several physical parameters.	physical-level
Chip-level Soft Error Estimation [25]	The authors present a method to estimate the failure of a full-chip. It is based on timing analyses of basic devices. A combination of these analyses helps build the final full-chip failure rate.	physical-level and system-level simulation
Monte-Carlo Simulation [10], [26]	A random error injection and function verification to measure the reliability of the system under failure.	system-level simulation
System Reliability Block Diagrams [10]	This model presents the logic relationship of the system's components. There are five basic systems: serial, parallel, stand-by, k-out-of-n and complex systems. All of these systems are analyzed using the probability theory.	analytical model
Fault-tree Analysis [10]	A graphical and logical representation of possible events occurring in a system.	analytical model
Markov Model [10], [27]	It models the system's possible failure situations as a set of states. There are possible failure rates and repair rates between two states. The final reliability of the system is obtained by calculating the probability of the dead-end states.	analytical model
TSV Failure Analysis [17]	This analytical model analyzes the probability of failure of 3D-NoCs under the failure of TSVs. In 3D-NoCs, TSVs are used as the vertical connection wires.	analytical model (for NoCs)
NoCs Reliability Analysis [18]	This model estimates the reliability of a NoC system using probability theory.	analytical model (for NoCs)

reliability block diagrams where the system is modeled in logic relationships and the results are obtained by using probability theories. To reduce the system complexity, the cut-sets method can calculate the subsystem and merge them to the final system. The fault-tree analysis exploits the operation of a given design to obtain the possible events of the system. The above-mentioned methods have a common drawback, which manifests in their lack of flexibility and reconfigurability when it comes to reconfigurable and repairable system. This can be dealt with the Markov model, which analyzes the system's events, configurations, and behaviors as states and builds a graph-based model. After that, the reliability of the system can be obtained. Analytical models for NoCs are recently presented [37]–[40]. A reliability assessment for TSV failure in 3-D-NoCs is addressed in [17] and [41]. The reliability of an NoC is also analyzed in [18]. These methods have provided promising solutions for NoCs' reliability assessment; however, they lack the support for FT and highly complex systems.

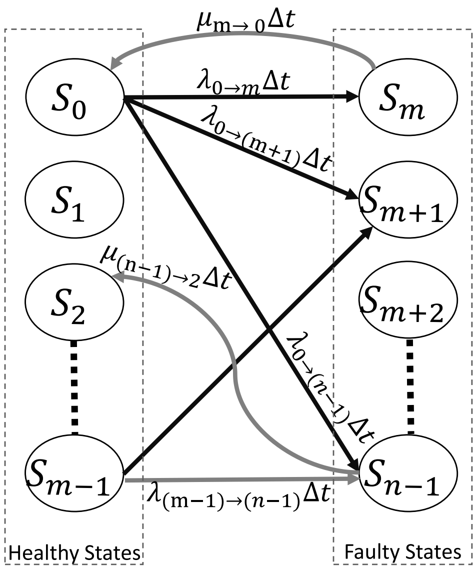
$\mu_{m\to 0}\Delta t$
S_0
$\lambda_{0\to m}\Delta t$
S_m
$\lambda_{0\to(m+1)}\Delta t$
S_1
S_{m+1}
$\mu_{(n-1)\to 2}\Delta t$
S_2
S_{m+2}
$\lambda_{0\to(n-1)}\Delta t$
S_{m-1}
S_{n-1}
$\lambda_{(m-1)\to(n-1)}\Delta t$
Healthy States
Faulty States

Fig. 5. Markov-state reliability model for an n states system with m nonfaulty states.

IV. MARKOV-STATE MODEL AND ASSESSMENT DEFINITIONS

This section presents the basic concepts, definitions, and assumptions used in the proposed analytical assessment methodology. Specifically, we adopt the Markov-state model to analyze the reliability of the fault-tolerance mechanisms. Therefore, we first start by giving an overview of the Markov-state model followed by the different assumptions and models adopted in the proposed assessment method.

A. Markov State Model Overview

A system operating with faults can be converted into a Markov state model [27], [42], as shown in Fig. 5. Each state S_i of the Markov model represents a possible status (event, configuration, and behavior) of the system. A status can be a case where one or multiple elements of the system fail. If the system's operation in state S_i is maintained correctly, we define S_i as "healthy." If the system is unable to operate correctly in this state, we define it as "faulty."

The reliability of a system can be defined as a time-dependent probability function $R(t)$ in the time domain [$R^*(s)$ in Laplace domain] and can be evaluated using the MTTF [27] calculation as follows:

$$\text{MTTF} = \int_{t=0}^{\infty} R(t) = \lim_{s\to 0}(R^*(s)). \quad (1)$$

We assume that the system's status is converted to a set named $\mathbb{S}$ with n states: $S_0 \ldots S_{n-1}$. We use S_0 to represent the initial state of the system. The set of healthy states and the set of faulty states are defined, respectively, as follows:

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

and

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

Fig. 5 shows each state S_i of $\mathbb{S}$. A transition between two states has a specific rate, which is defined as follows.

1) λ is the fault rate of a component in the system. It can represent a transition from an element of set $\mathbb{H}$ to an element of set $\mathbb{F}$.
2) μ is the repair rate of a component in the system. It can represent a transition from an element of set $\mathbb{F}$ to an element of set $\mathbb{H}$.

The reliability of a system can be obtained by summarizing the probabilities of its states and the reliability function. Given a state S_i, which has j input transitions (σ) from j states $S_{n,i}$ ($n = 1, 2, \ldots, j$) and k output transitions (γ) to k states $S_{m,i}$ ($m = 1, 2, \ldots, k$), the derivative of the probability of the state in the time domain is given as follows:

$$p'_{S_i}(t) = -\sum_{n=1}^{j} \sigma_n p_{S_{n,i}} + \sum_{m=1}^{k} \gamma_m p_{S_{m,i}}. \tag{4}$$

Note that with each input or output transition, based on the set of the state, the transition rates (σ and γ) can be either a failure (λ) or repair (μ) transition. By converting (4) to the Laplace domain [27], we obtain the equation as follows:

$$sP_{S_i}(s) - p_{S_i}(0) = -\sum_{n=1}^{j} \sigma_n P_{S_{n,i}} + \sum_{m=1}^{k} \gamma_m P_{S_{m,i}}. \tag{5}$$

When applying the above for all states, we obtain m equations of $(P_{S_0}, P_{S_1}, \ldots, P_{S_{m-1}})$ and the reliability function $R^*(s)$ can be defined as the sum of probabilities of being in *healthy* states

$$R^*(s) = P(\mathbb{H}) = \sum_{S_i \in \mathbb{H}} P(S_i). \tag{6}$$

Finally, to obtain the MTTF value, (1) can be used.

B. Assumptions

In an FT system, the FT method has to improve the reliability of the system. As a result, the MTTF value has to be also increased. Therefore, we propose the reliability acceleration factor (RAF) to denote the efficiency of fault tolerance, represented as

$$\text{RAF} = \frac{\lambda_{\text{original}}}{\lambda_{\text{FT}}} = \frac{\text{MTTF}_{\text{FT}}}{\text{MTTF}_{\text{original}}} \geq 1 \tag{7}$$

where the following hold.

1) $\text{MTTF}_{\text{original}}$ is the MTTF of the original system.
2) MTTF_{FT} is the MTTF of the FT system.
3) λ is the fault rate and it is the inverse value of MTTF.

The transitions have dedicated "steady-state" rates, which need to be predefined [27]. Therefore, we make the following assumptions regarding a given system failure.

1) The system starts with a default state where all components are in the healthy state. In Fig. 5, the initial status is: $p_{S_0}(0) = 1$ and $p_{S_i}(0) = 0$ with $i \neq 0$.
2) The failure rates are constant.

Since the fault rate depends on the technology parameters, running environment and operating circumstances are not easy to obtain in the early analysis. Therefore, we assume the "raw" fault rate (i.e., original fault rate with no fault-tolerance support) according to the following assumptions.

1) The fault rate of a module has a linear relationship with its area cost.
2) The fault rate of a module has a linear relationship with its operating time.
3) The fault rate is affected after a module is attached to a system.

Thus, for a system with k components, its fault rate is given by

$$\lambda_{\text{system}} = \frac{1}{\text{MTTF}_{\text{system}}} = \sum_{i=1}^{k} f_i \pi_i \lambda_{\text{unit}} \tag{8}$$

where unit is a selected module as a reference for calculation. π_i is the fault-rate ratio between the component and the unit. It can be defined as the area cost ratio and operating time ratio [12], [35]. f_i is the fault-rate ratio after attaching the component to the system (f=1/RAF). The fault rate can be reduced ($f_i < 1$) by applying fault tolerance; otherwise, it remains as $f_i = 1$.

In a typical FT system, a faulty part can be repaired with a specific repair rate (μ) after being detected. This rate is given by the module managing the fault-tolerance mechanism.

C. Classified Model

We categorize fault-tolerance architectures [5], [9], [35] into four basic models where each model is treated separately and differently: non-FT model, spare model, fault-reduced model, and error handling model as follows.

1) *Model 0 (Non-FT Model):* This model is applied for the module without fault-tolerance capabilities. Its fault rate can be obtained by (8) or based on physical-level analysis.
2) *Model 1 (Spare Model):* As represented in Fig. 6(a), we assume that the considered module has m separated identical parts, which can function with at least n parts. In the redundancy method, an r extra spare parts are added in the design stage. The self-configuration (previously presented in Section II-C) can be modeled without extra parts. In fact, it has $n < m$ and allows the system to fail at most $(m - n)$ submodules.
3) *Model 2 (Fault-Reduced Model):* This model is aimed for fault-reduced systems. The reducing of fault rate can be given by a special technique (e.g., error correcting code [30]). This model can help applying the other former analyses (physical level or system level) to the new system.
4) *Model 3 (Error Handling Model):* As shown in Fig. 6(b), this model is designed for error detection and management modules. The detection module also adds a new

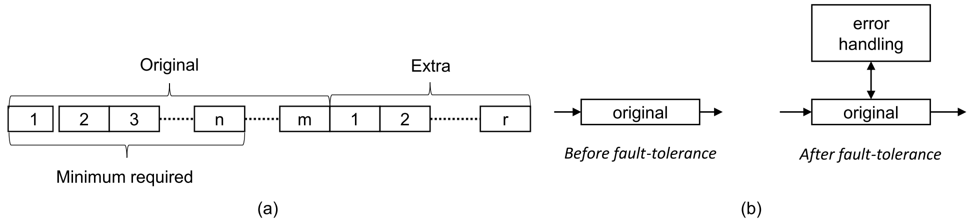

Fig. 6. Classified model. (a) Model 1—spare. (b) Model 3—error handling.

Dividing:

1) A Network-on-Chip consists of N_R routers.
2) A router is divided into several modules: $M_0, M_1, ..., M_a$.
3) Each module can be modeled as one of the predefined models (Model 0-3).

Conquering:

1) For each module $M_i = M_0, M_1, ..., M_a$ of a router, analyze it using one of the given strategies:
   - If the module M_i is not fault tolerant (Model 0), it is given reliability values by Eq. 8.
   - If the module M_i is a spare model or a reconfigurable module (Model 1), it is given a failure rate by *Strategy 1*.
   - If the module M_i is reduced failure by applying a special technique (Model 2), use *Strategy 2*.
   - If the module M_i is a typical fault-tolerant module (Model 3), after applying *Strategy 1 or 2* its final failure rate is given by calculation in *Strategy 3*.

Merging:

1) A router reliability is obtained by *Router Merging*.
2) A network reliability is obtained by *Network Merging*.

Fig. 7. Reliability assessments for FT NoC.

rate to the overall system. The fault rate of the original module can be reduced by using Model 1 or Model 2.

V. Quantitative Reliability Assessment

In this section, we present a detailed explanation on how to evaluate the reliability of the different components that can constitute an FT NoC system. Fig. 7 shows the three main steps that are necessary to obtain a comprehensive reliability assessment. A network is divided into routers, which are divided into modules (*Dividing*). After dividing, the modules are analyzed and classified according to their appropriate model, and the suitable strategy is applied (*Conquering*). The final reliability is obtained by merging all modules together (*Merging*).

A. Conquering

We consider the components of a router by using the following strategies, which are applied to each one of the four basic models presented in Section IV-C.

1) Strategy 0 *(Applied for Model 0—Non-FT Model):* If the module is not FT, its failure rate is simply estimated using (8).

2) Strategy 1 *(Applied for Model 1—Spare Model):* This strategy handles hard faults using spare modules or by reconfiguring an alternative part.

We assume that the considered module has m separate identical parts and can function with at least n parts. In order to enhance the reliability, extra r spare parts are added in the design stage. f is the number of parts that are faulty in a state. Equation (2) can be then reformulated as

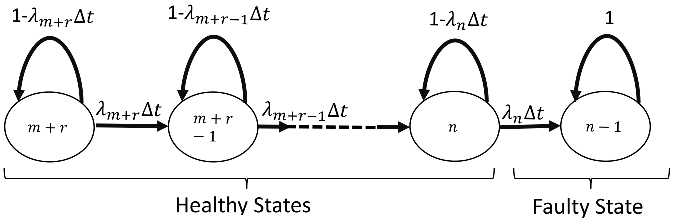

Fig. 8. Markov-state reliability model for spare modules.

$$\mathbb{H} \triangleq \{S_i \in \mathbb{S} | m + r - f \geq n\}. \tag{9}$$

The Markov state model can be built, as shown in Fig. 8. Each state is labeled with the number of healthy parts and the failure rate is indicated by (8).

The original system consists of m parts, and its MTTF can be expressed as

$$\text{MTTF}_{\text{original}} = \frac{1}{m}\frac{1}{\lambda_{\text{single-part}}}. \tag{10}$$

By applying (1) based on the probability of healthy states, the MTTF can be expressed as

$$\text{MTTF}_{\text{FT}} = \Sigma_{i=n}^{m+r}\frac{1}{\lambda_i} = \frac{1}{\lambda_{\text{single-part}}}\left(\Sigma_{i=n}^{m-1}\frac{1}{i} + \frac{1}{m} + \Sigma_{i=m+1}^{m+r}\frac{1}{i}\right). \tag{11}$$

Lemma 1: The RAF values can be calculated as follows:

$$\begin{aligned}\mathsf{RAF}_{\text{model}-1} &= \frac{\text{MTTF}_{\text{FT}}}{\text{MTTF}_{\text{original}}} = \Sigma_{i=n}^{m+r}\frac{m}{i}\\ &= 1 + \Sigma_{i=n}^{m-1}\frac{m}{i} + \Sigma_{i=m+1}^{m+r}\frac{m}{i}.\end{aligned} \tag{12}$$

Proof 1: We consider a system originally having *m* identical parts, *r* extra parts, and requires at least *n* parts for maintaining its function. The derivatives of probabilities of each state in time domain are calculated as follows:

$$\begin{aligned}p'_{m+r} &= -\lambda_{m+r}p_{m+r}\\ p'_{m+r-1} &= \lambda_{m+r}p_{m+r} - \lambda_{m+r-1}p_{m+r-1}\\ p'_{m+r-2} &= \lambda_{m+r-1}p_{m+r-1} - \lambda_{m+r-2}p_{m+r-2}\\ &\dots\\ p'_{n} &= \lambda_{n+1}p_{n+1} - \lambda_{n}p_{n}.\end{aligned} \tag{13}$$

By converting to Laplace domain, the probabilities of states are expressed as

$$\begin{aligned}sP_{m+r} - p_{m+r}(0) &= -\lambda_{m+r}P_{m+r}\\ sP_{m+r-1} - p_{m+r-1}(0) &= \lambda_{m+r}P_{m+r} - \lambda_{m+r-1}P_{m+r-1}\\ sP_{m+r-2} - p_{m+r-2}(0) &= \lambda_{m+r-1}P_{m+r-1} - \lambda_{m+r-2}P_{m+r-2}\\ &\dots\\ sP_{n} - p_{n}(0) &= \lambda_{n+1}P_{n+1} - \lambda_{n}P_{n}.\end{aligned} \tag{14}$$

By resolving the above-mentioned equations, the final probabilities in Laplace domain are

$$\begin{aligned}P_{m+r} &= \frac{1}{s+\lambda_{m+r}}\\ P_{m+r-1} &= \frac{\lambda_{m+r}P_{m+r}}{s+\lambda_{m+r-1}} = \frac{\lambda_{m+r}}{(s+\lambda_{m+r})(s+\lambda_{m+r-1})}\\ P_{m+r-2} &= \frac{\lambda_{m+r-1}P_{m+r-1}}{s+\lambda_{m+r-2}}\\ &= \frac{\lambda_{m+r}\lambda_{m+r-1}}{(s+\lambda_{m+r})(s+\lambda_{m+r-1})(s+\lambda_{m+r-2})}\\ &\dots\\ P_{n} &= \frac{\lambda_{n+1}P_{n+1}}{s+\lambda_{n}} = \frac{\lambda_{m+r}\dots\lambda_{n+1}}{(s+\lambda_{m+r})(s+\lambda_{m+r-1})\dots(s+\lambda_{n})}.\end{aligned} \tag{15}$$

The reliability of the system is given by the healthy states as follows:

$$\begin{aligned}R^{*}(s) &= P(\mathbb{H}) = \Sigma_{i=n}^{m+r}P_i\\ \text{MTTF}_{\text{FT}} &= \lim_{s\to 0}(R^{*}(s)) = \lim_{s\to 0}\Sigma_{i=n}^{m+r}P_i\\ \text{MTTF}_{\text{FT}} &= \Sigma_{i=n}^{m+r}\frac{1}{\lambda_i} = \frac{1}{\lambda_{\text{single}-\text{part}}}\Sigma_{i=n}^{m+r}\frac{1}{i}\\ &= \frac{1}{\lambda_{\text{single}-\text{part}}}\left(\Sigma_{i=n}^{m-1}\frac{1}{i} + \frac{1}{m} + \Sigma_{i=m+1}^{m+r}\frac{1}{i}\right).\end{aligned} \tag{16}$$

Finally, RAF can be calculated, using (10) and (16), as follows:

$$\mathsf{RAF}_{\text{FT}} = \frac{\text{MTTF}_{\text{FT}}}{\text{MTTF}_{\text{original}}} = 1 + \frac{\Sigma_{i=n}^{m-1}\frac{1}{i} + \Sigma_{i=m+1}^{m+r}\frac{1}{i}}{\frac{1}{m}}. \tag{17}$$

When simplifying (17), (12) can be obtained. The enhancement of reliability is obtained thanks to the additional extra parts (r) and the reduction of the minimal required part (*n*).

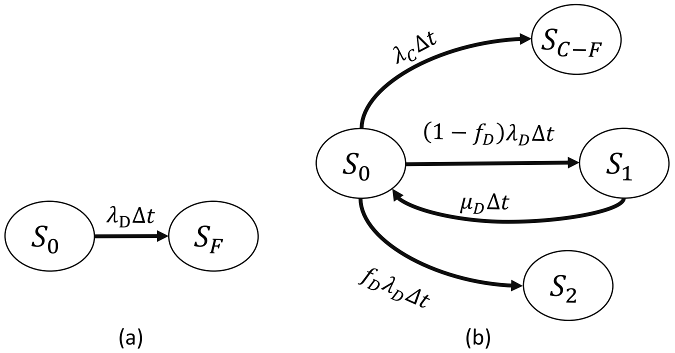

Fig. 9. Simplified Markov-state reliability model for (a) original system and (b) FT system.

3) Strategy 2*:* This strategy is designed for Model 2—fault-reduced model. In this case, the efficiency can be predicted from other analyses or probability calculations. With a fault reduction value f_{FT} given by the technique, the new fault rate is obtained by (18)

$$\lambda_{\text{FT}} = f_{\text{FT}}\lambda_{\text{original}} \tag{18}$$

where f_{FT} is the inverse value of RAF

$$f_{\text{FT}} = \frac{1}{\text{RAF}} = \frac{\lambda_{\text{FT}}}{\lambda_{\text{original}}}. \tag{19}$$

This strategy is proposed to help designers integrate the existing analyses into our method. For instance, f_{FT} can be obtained from a physical-level analysis, a simulation, another analytical model, or even from the FAIT stage. By dividing the fault rate (or MTTF) of design before and after applying the fault-tolerance methods, designers can obtain the reduction rate and integrate the fault-reduced module to the system.

4) Strategy 3*:* This strategy is applied to Model 3—error handling model. As previously mentioned, in prior strategies, we demonstrate the efficiency of the fault-tolerance techniques using analytical models [43]. Because fault-tolerance requires additional modules for checking and correcting faults, these additional modules also add additional fault rates.

We model both original and FT systems to have two Markov state models as represented in Fig. 9(a) and (b), respectively, where the following hold.

1) S_0 is the initial state.
2) S_F is the faulty state of the original system.
3) S_1 is the faulty state of the original system, which can be corrected by the FT technique.
4) S_2 is the faulty state of the original system, which cannot be corrected by the FT technique.
5) S_{C-F} is the faulty state of the repair module.

Because of the protection from the FT technique, the FT system can handle some faults. Therefore, we define the transition rates as follows.

1) λ_D is the fault rate of the original system (D).
2) λ_C is the fault rate of the repair module of the FT system.
3) μ_D is the repair rate, which is provided by the repair module (C) on the original system (D).

4) f_D is the fault reducing value by applying the fault-tolerance mechanism.

Based on the Markov state of the two systems, as shown in Fig. 9, the reliability function is givens as

$$R^*(s) = P_{S0}. \tag{20}$$

The final fault rate of the fault-tolerance system is as follows:

$$\lambda_{\mathrm{FT}} = f_D\lambda_D + \lambda_C. \tag{21}$$

Lemma 2: The RAF value can then be expressed as

$$\mathsf{RAF}_{\mathrm{FT}} = f_D + \frac{\lambda_C}{\lambda_D}. \tag{22}$$

Proof 2: We have the derivations of the states as the following:

$$\begin{aligned} p'_{S0} &= -(\lambda_C + (1 - f_D + f_D)\lambda_D)p_{S0} + \mu_D p_{S1} \\ p'_{S1} &= -\mu_D p_{S1} + (1 - f_D)\lambda_D p_{S0} \\ p'_{C-F} &= (\lambda_C)p_{S0} \\ p'_{S2} &= (f_D\lambda_D)p_{S0}. \end{aligned} \tag{23}$$

And their Laplace transforms can be expressed as

$$\begin{aligned} sP_{S0} - p_{S0}(0) &= -(\lambda_C + \lambda_D)P_{S0} + \mu_D P_{S1} \\ sP_{S1} - p_{S1}(0) &= -\mu_D P_{S1} + (1 - f_D)\lambda_D P_{S0} \\ sP_{C-F} - p_{C-F}(0) &= (\lambda_C)P_{S0} \\ sP_{S2} - p_{S2}(0) &= (f_D\lambda_D)P_{S0} \end{aligned} \tag{24}$$

$$\begin{aligned} sP_{S0} - 1 &= -(\lambda_C + \lambda_D)P_{S0} + \mu_D P_{S1} \\ sP_{S1} - 0 &= -\mu_D P_{S1} + (1 - f_D)\lambda_D P_{S0} \\ sP_{C-F} - 0 &= (\lambda_C)P_{S0} \\ sP_{S2} - 0 &= (f_D\lambda_D)P_{S0} \end{aligned} \tag{25}$$

$$\begin{aligned} sP_{S0} + (\lambda_C + \lambda_D)P_{S0} &= 1 + \mu_D P_{S1} \\ sP_{S1} + \mu_D P_{S1} &= (1 - f_D)\lambda_D P_{S0} \\ sP_{C-F} &= (\lambda_C)P_{S0} \\ sP_{S2} &= (f_D\lambda_D)P_{S0} \end{aligned} \tag{26}$$

$$\begin{aligned} P_{S0} &= \frac{1 + \mu_D P_{S1}}{s + (\lambda_C + \lambda_D)} \\ P_{S1} &= \frac{(1 - f_D)\lambda_D P_{S0}}{s + \mu_D} \\ P_{C-F} &= \frac{\lambda_C}{s}P_{S0} \\ P_{S2} &= \frac{f_D\lambda_D}{s}P_{S0} \end{aligned} \tag{27}$$

$$\begin{aligned} P_{S0} &= \frac{1 + \mu_D\dfrac{(1 - f_D)\lambda_D P_{S0}}{s + \mu_D}}{s + (\lambda_C + \lambda_D)} \\ P_{S0}\left(1 - \frac{\mu_D(1 - f_D)\lambda_D}{(s + \lambda_C + \lambda_D)(s + \mu_D)}\right) &= \frac{1}{s + \lambda_C + \lambda_D} \\ P_{S0}((s + \lambda_C + \lambda_D)(s + \mu_D) - \mu_D(1 - f_D)\lambda_D) &= (s + \mu_D) \\ P_{S0} &= \frac{s + \mu_D}{s^2 + (\lambda_C + \lambda_D + \mu_D)s + \mu_D(\lambda_C + f_D\lambda_D)} \end{aligned} \tag{28}$$

$$\begin{aligned} P_{S0} &= \frac{s + \mu_D}{s^2 + (\lambda_C + \lambda_D + \mu_D)s + \mu_D(\lambda_C + f_D\lambda_D)} \\ P_{S1} &= \frac{(1 - f_D)\lambda_D}{s^2 + (\lambda_C + \lambda_D + \mu_D)s + \mu_D(\lambda_C + f_D\lambda_D)} \\ P_{C-F} &= \frac{\lambda_C(s + \mu_D)}{s^2 + (s(\lambda_C + \lambda_D + \mu_D)s + \mu_D(\lambda_C + f_D\lambda_D))} \\ P_{S2} &= \frac{f_D\lambda_D(s + \mu_D)}{s(s^2 + (\lambda_C + \lambda_D + \mu_D)s + \mu_D(\lambda_C + f_D\lambda_D))}. \end{aligned} \tag{29}$$

The MTTF of the final system is given by the healthy state *S0*

$$\begin{aligned} \mathrm{MTTF}_{\mathrm{FT}} &= \lim_{s\to 0}(R^*(s)) = \lim_{s\to 0} P_{S0} \\ \mathrm{MTTF}_{\mathrm{FT}} &= \frac{1}{(\lambda_C + f_D\lambda_D)}. \end{aligned} \tag{30}$$

Because the MTTF value of the original system is $1/\lambda_D$, the RAF of this model is given as

$$\mathsf{RAF}_{\mathrm{FT}} = \frac{\mathrm{MTTF}_{\mathrm{FT}}}{\mathrm{MTTF}_{\mathrm{original}}} = \frac{\lambda_C + f_D\lambda_D}{\lambda_D}. \tag{31}$$

When simplifying the above-mentioned equation, (22) can be obtained.

B. Merging

After analyzing all modules, the system reliability is calculated based on its submodules. Therefore, we first start by merging the router components and then merge all the network components to obtain the entire system reliability.

1) Router Merging: We first determine that the router is reliable only if it is able to transmit correctly a given data from any input to any output port. By applying (8), the fault rate of a router is obtained as follows:

$$\lambda^*_{\mathrm{router}} = \sum_{i=1}^{N} f_{M_i}\lambda_{M_i} \tag{32}$$

where λ_{M_i} is the fault rate of module M_i and f_{M_i} is the fault reduction rate given by attaching this module to the system. By applying (8) with unit, which is defined as a baseline router, the new failure rate of a router is given as follows:

$$\lambda^*_{\mathrm{router}} = \sum_{i=1}^{N} f_{M_i}\pi_{M_i}\lambda_{\mathrm{router}}. \tag{33}$$

The RAF value can also be obtained by the following equation:

$$\mathrm{RAF}_{\mathrm{router}} = \frac{\lambda^*_{\mathrm{router}}}{\lambda_{\mathrm{router}}} = \sum_{i=1}^{N} f_{M_i}\pi_{M_i}. \tag{34}$$

2) Network Merging: The next step is to evaluate the network reliability. Unlike the router, the network has a high flexibility in the routing process. For example, if a link along the path between two distant routers is faulty, the network can avoid it in the routing path.

Among several existing network's reliability terminologies [27], this paper uses "all-terminal reliability" for the analysis. "All-terminal reliability" is defined, as all PEs are

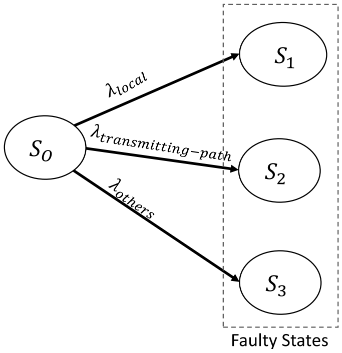

Fig. 10. Markov state of a mesh-based network.

connected to the network. In other words, all PEs have the ability to communicate with any PE in the network. According to [27] and [33], the most common method is using node-based reliability. Unlike computer or transportation network, NoCs have a small granularity in terms of fault occurrence and handling (first in first out, wires, and logic circuit). Therefore, a low-level approach is more suitable. In this assessment, we analyze the network reliability in terms of connection between two routers, or a pair of router and PE.

To analyze the network's reliability, we analyze the possible failure cases that may corrupt the system. We define the major failure cases as follows.

1) *Failure on Local Connection:* A failure on the connections, which are handled by NIs, between routers and PEs can corrupt the network's reliability. This involves two channels (input and output) and an input buffer, which is constantly attached to its input channel.
2) *Failure on Transmitting Path:* A failure on the transmitting path can corrupt the network's reliability. The transmitting path is considered as a set of connections between routers.
3) *Failed on Other Router Modules:* A failure in nontransmitting parts (e.g., switch allocator and management modules) of a router may malfunction the router.

From the failure cases, a Markov state model is built, as shown in Fig. 10. As a result, the fault rate of a network of N_R routers is given as follows:

$$\lambda_{\text{network}} = \lambda_{\text{local}} + \lambda_{\text{transmitting-path}} + \lambda_{\text{others}} \tag{35}$$

where the following hold.

1) $\lambda_{\text{local}} = N_R \times (2\lambda_{1-\text{channel}} + \lambda_{\text{input-buffer}})$ is the fault rate of all local connections. N_R is the number of routers in the network.
2) $\lambda_{\text{transmitting-path}}$ is the fault rate of transmitting paths between routers inside the network. Designers can estimate this value based on analytical analysis or simulation model proposed in [10], [27], [33], and [44]. Here, we apply *k-failure* [44] model to assess the reliability.
3) λ_{others} is given by the fault rates of other parts (nonrouting parts) of routers.

In this paper, we consider $\lambda_{\text{transmitting-path}} = \lambda_{\text{RTR}} \times N_R$. λ_{RTR} is the fault rate of a router-to-router (RTR) connection, which represents one node connection from a router to any adjacent router. The RTR connection failure rate depends on the position of the router in the network. Here, we use the k-failure [44] model: a router is disconnected at the presence of k-failures. We adopted this model with a modification: the failure value k is defined as a connection and it depends on the router's position. For example, the corners, the edges, the side, and the middles of the 3-D mesh NoCs have three, four, five, and six routing selections, respectively. This condition is the maximum value that FT routing algorithms can achieve. With the fault assumption in (8), the routers located at a similar position (corner, edge, side, or middle) have a similar fault rate. Therefore, the failure rate of RTR connection is expressed as follows:

$$\begin{aligned}\lambda_{\text{RTR}} = P_{\text{corner}} \times \lambda_{\text{corner}} + P_{\text{edge}} \times \lambda_{\text{edge}} \\ + P_{\text{side}} \times \lambda_{\text{side}} + P_{\text{middle}} \times \lambda_{\text{middle}}\end{aligned} \tag{36}$$

where the failure rates: λ_{corner}, λ_{edge}, λ_{side}, and λ_{middle} are obtained from the fault rate of a connection. P_{corner}, P_{edge}, P_{side}, and P_{middle} are the probability of having a router in corner, edge, side, and middle of the network, respectively. A connection is defined as a data path from input buffer to the next input buffer or NI's buffer. As shown in Fig. 1, a connection consists of an input buffer, a crossbar link, and an interrouter channel. Because these components are independent, the fault rate of a connection ($\lambda_{\text{conn.}}$) is defined as follows:

$$\lambda_{\text{conn.}} = \lambda_{1-\text{input-buffer}} + \lambda_{1-\text{crossbar-link}} + \lambda_{1-\text{router-channel}}. \tag{37}$$

In order to compute λ_{RTR}, the fault rate of each position in (36) can be calculated by using (11) of Strategy 1 as follows:

$$\lambda_{\text{position}} = \frac{1}{\text{MTTF}_{\text{position}}} = \frac{1}{\sum_{i=n}^{m+r} \frac{1}{\lambda_{\text{conn.}}}} \tag{38}$$

where the identical part is a connection (its fault rate is $\lambda_{\text{conn.}}$), $r = 0$, $n = 1$, and $m = 3$, 4, 5, and 6 for the corner, edge, side, and middle, respectively. The non-FT routing fault rate (from MTTF) can be calculated using (10).

In the case where the routers in a network have different fault rates, and despite having the same architecture, we need to manually calculate each router's $\lambda_{\text{RTR}}(i = 1, 2, \ldots, N_R)$. The fault rate of a transmitting path is obtained by the following equation:

$$\lambda_{\text{transmitting-path}} = \sum_{i=1}^{N_R} \lambda_{\text{RTR}}(i) \tag{39}$$

where $\lambda_{\text{RTR}}(i)$ is the fault rate of RTR connection from router i of the network.

VI. Evaluation Results

A. Evaluation Methodology

In this section, we evaluate the reliability of NoC systems with two methodologies: the proposed analytical model and a system-level simulation. To demonstrate the efficiency of the proposed analytical analysis, we used our previously proposed 3-D NoC system [45], called 3-D hard-fault soft-error tolerant OASIS (FETO) as a case of study. The 3-D-FETO uses the

TABLE II
SIMULATION CONFIGURATIONS

Parameters	Value
Flits Size	44 bits
Header Size	14 bits
Buffer Depth	4
Switching	Wormhole-like
Flow-control	Stop-Go
Routing	Look-ahead Fault-Tolerant

Wormhole-like switching technique [46] with the ability to choose between Wormhole and Store-and-Forward switching depending on the level of packet fragmentation. There are three main techniques on hard fault tolerance [47]: 1) random access buffer (RAB) that isolates the faulty buffer from writing and reading processes; 2) bypass-link-on-demand (BLoD) that uses two backup links to handle faults in crossbar; and 3) look-ahead-FT (LAFT) routing algorithm to avoid faulty links by selecting an alternative path. There are also two main techniques used for soft-error resilience: 1) pipeline computation redundancy [48] that reexecutes the routing and arbitration processes to detect soft errors and handle them and 2) SECDED error correcting code [30] that is selected to deal with bit flip. For the detection mechanism, we use an online detection scheme [45], which can detect failures in input buffers, crossbar, and links to provide an appropriate solution (RAB, BLoD, or LAFT). Please note that the proposed methodology is valid for any NoC system regardless of the configuration adopted, and the 3-D-FETO is used only as an example.

The 3-D-FETO NoC system was designed in Verilog HDL, synthesized, and prototyped with commercial computer-aided design (CAD) tools and VLSI technology, respectively [49], [50]. We select a baseline NoC model (OASIS) [46] in these evaluations. To compare between the two methods, we use a similar error rate and then calculate the MTTF and RAF values. The configurations are shown in Table II. The final system-level simulation results are compared with the analytical model results to study the accuracy of the proposed model.

B. MTTF Monte Carlo Simulation

In order to illustrate the accuracy of the proposed assessment method, we compare it with a Monte Carlo simulation [10]. We use the fault injection method on netlist files [26], [51] for more accurate results. Based on the same fault assumptions as the assessment, we also calculate the MTTF and RAF values for the ease of comparison. Fig. 11 shows the MTTF Monte Carlo setting up flow [45]. The goal of this simulation is to measure the MTTF value of the system. The flow of this simulation consists of the following.

1) The NoC architecture is designed in Verilog HDL and synthesized using the Synopsys Design Compiler to obtain a postsynthesis netlist model.
2) A fault distribution system is automatically integrated inside the NoC netlist model by using Python's Regular Expression scripts [52]. The faults are modeled in stuck at "0" and "1." Our method is similar to the fault injection methods in [26] and [51].

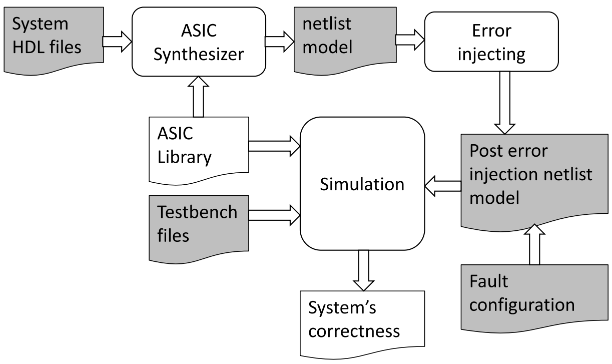

Fig. 11. Monte Carlo setting up flow.

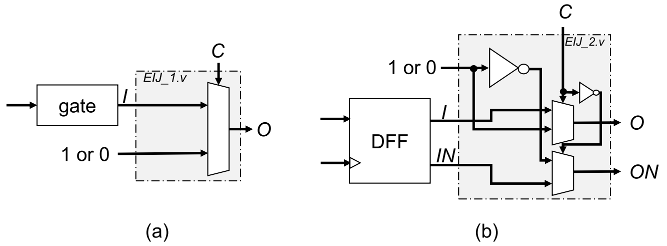

Fig. 12. Error injector architecture. (a) Single output gate. (b) Flip-flop with two outputs.

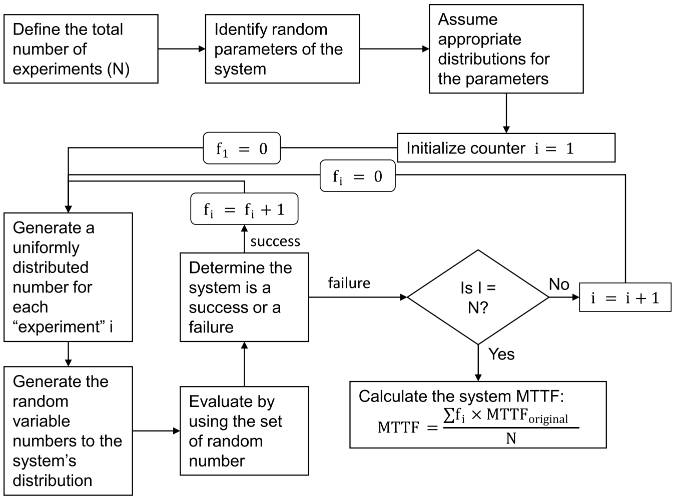
Define the total number of experiments (N)
Identify random parameters of the system
Assume appropriate distributions for the parameters
Initialize counter i = 1
f1 = 0
fi = 0
fi = fi + 1
success
Generate a uniformly distributed number for each "experiment" i
Determine the system is a success or a failure
failure
Is I = N?
No
i = i + 1
Yes
Generate the random variable numbers to the system's distribution
Evaluate by using the set of random number
Calculate the system MTTF:
MTTF = ∑fi × MTTForiginal / N

Fig. 13. MTTF Monte Carlo simulation process.

3) The posterror injection netlist model is used to simulate and find the number of faults leading to failure.
4) The number of faults is recorded for further processing.

The error injector architectures are shown in Fig. 12 with two models: normal gate and flip-flop gate. When the error injector is enabled (by setting up $C = 1$), it forces the output of its attached gate to "0" or "1." If the error injector is disabled, the correct output is forwarded.

The simulation process is shown in Fig. 13.

1) At the first iteration, the injected position is generated and the corresponding position is distributed with a fault.
2) Hard faults are injected until finishing the experiment. Soft errors disappear after one clock cycle.
3) A test bench is designed to verify the system correctness after injecting a fault (100% of the PEs are connected

TABLE III
ROUTER'S WEIGHT AND GATE RATIO

Module	Submodule	Weight	Gate Ratio
Network	Network	100%	100%
	Routers	70%	100%
	Channels	30%	0%
Router	Router	100%	100%
	Input Buffer	69.72%	7.90%
	Crossbar	8.00%	11.43%
	Switch-Allocator	7.00%	16.97%
	Others	15.28%	63.7%

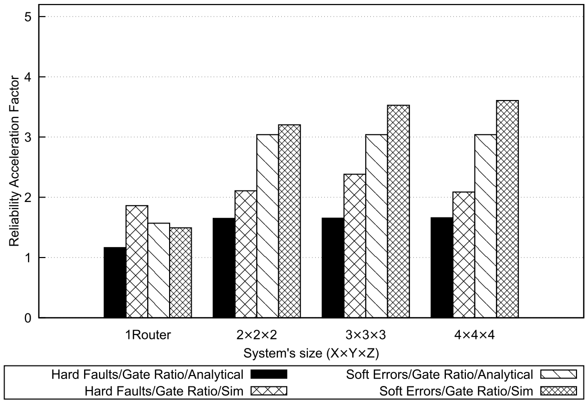

Fig. 14. Comparison results with gate ratio distributions.

and 100% of the packets are correctly delivered). If the system is still functioning correctly after a fault is injected, the simulation injects more faults. Otherwise, the number of faults and working time is recorded.

4) The time-to-failure is calculated based on the number of faults or the working time. The final MTTF is the average value of all iterations.

For iteration i, the time-to-failure is measured as the number of faults injected as

$$\mathrm{TTF}_i = f_i \times \frac{1}{\lambda_{\mathrm{system}}} \tag{40}$$

where $\lambda_{\mathrm{system}}$ is the raw fault rate of the original system. After finishing a simulation of N iterations, the MTTF value of a system is given by (41), where f_i is the number of faults causing failure in experiment i

$$\mathrm{MTTF}_{\mathrm{system}} = \frac{\sum_{i=1}^{N} \mathrm{TTF}_i}{N} = \frac{\sum_{i=1}^{N} f_i}{N} \times \frac{1}{\lambda_{\mathrm{system}}}. \tag{41}$$

In order to verify the analytical model, we use two configurations: 1) gate ratio—where the fault rate is linear to the number of utilized gates in the netlist file and 2) weight—where the fault rate is higher in the data transmission modules, which are likely to have a significant impact on the system correctness. The fault-rate ratio can be seen in Table III where the errors are focused on input buffers, crossbar, and links.

Figs. 14 and 15 show the comparison of RAF values between the proposed analytical model and the simulation results. For a single router assessment, the analytical method predicts the RAF values with acceptable deviations (under 33%). Moreover, there is a significant amount of hidden faults in the baseline model. In this kind of situations, faults are injected; but, they do not give a detectable impact on the system. Especially, soft errors, which are injected in a single clock cycle, are likely to become hidden faults. In contrast, hard faults on the router give a higher impact on the system.

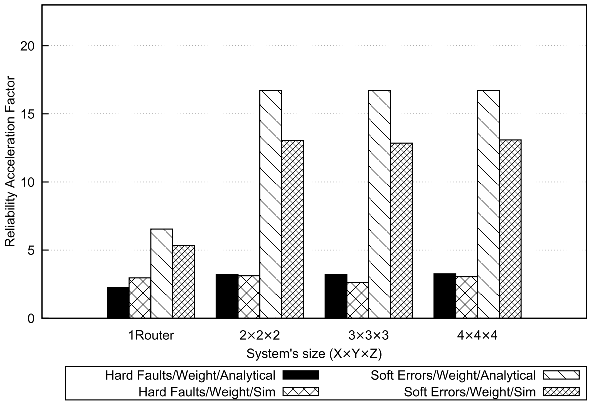

Fig. 15. Comparison results with weight distributions.

For networks' assessments, we also simulated and compared the RAF values for four different network sizes: $2 \times 2 \times 2$, $3 \times 3 \times 3$, and $4 \times 4 \times 4$. The worst cases can be observed in the soft-error simulation with weight distribution and hard-fault simulation with gate ratio distribution. However, the accuracy is still better than a single router where most of the deviations are less than 23%. The worst case is hard fault tolerance with the gate ratio distribution of a $3 \times 3 \times 3$ network where the difference can reach up to 31.64%. The deviation in the accuracy values is mostly caused by the occurrence of hidden faults. They are defined as the faults that can be injected without causing the system crash. Such hidden faults cause the observed difference between the analytical model and the Monte Carlo simulation. For instance, a fault on the routing unit or output port may lead to a misrouted packet. Nevertheless, at the next router, the packet can be routed correctly without causing deadlock or livelock in the network. Another example is the presence of hidden faults in the intrarouter routing where the employed routing algorithm chooses either the X-, Y-, or Z-direction depending on the used routing algorithm. At the presence of hidden faults, the order of dimensions might be altered. That is, instead of routing a packet through X and Y and then through Z, it is sent through Y and X and then through Z. However, and despite this change, the packet can still reach its destination correctly without the system crashing. Furthermore, and as shown in Fig. 15, the difference in soft errors is slightly higher than hard faults. This is because soft errors are injected within one clock cycle, which may not affect the operation of the system if this latter is idling. Because this type of faults did not crash the system, they make the non-FT system become more resilient and increase the MTTF values, while our model cannot estimate it. Since we adopted the weight distribution, where faults are injected more on the FT modules, the assessment results are closer than those of the gate ratio.

In summary, the overall accuracy is acceptable with most cases having less than 23% of deviations. There are some cases

TABLE IV
RELIABILITY ASSESSMENT SPEEDUP

Evaluated module	A MTTF simulation	Proposed method	Speedup
A router	11 hours	0.090 second	440,000
A $2 \times 2 \times 2$ network	20 hours	0.091 second	791,209
A $3 \times 3 \times 3$ network	2 days	0.092 second	1,878,261
A $4 \times 4 \times 4$ network	3.5 days	0.109 second	2,774,312

where the difference is considerable; however, the deviation is logical due to the low granularity of the reliability assessment. In fact, when designers apply the assessment method before obtaining the designs characteristics (e.g., gate ratios), the deviations are still reasonable.

C. Reliability Assessment Speedup

The proposed analytical method offers faster estimation time in comparison to other conventional methods. Table IV shows the reliability assessment simulation time and the speedup obtained with the proposed analytical model when compared with the conventional MTTF simulation, previously explained in Section VI-B. The proposed methods calculation is performed using GNU Octave, and the MTTF simulation performs netlist simulations for 1000 cases using the Cadence NCSim CAD tool. Both simulations were conducted on a Linux CentOS 6.4 machine using Intel Xeon E5-2620 (eight cores, 2.10 GHz) and 64 GB of RAM.

Thanks to the low complexity of the proposed method, the simulation time is always in the hundreds of milliseconds range for all cases. On the other hand, the MTTF simulation requires more than 11 h of computation for just a single router. This results in a speedup of 440 000 times with our proposed method. The long execution time of the MTTF simulation is caused by the main following factors: 1) the high complexity of the netlist files where a routers netlist file consists of over 15 000 separated gates; 2) the high complexity of the fault injection (i.e., each gate requires an error injection module and a fault distribution system); 3) the verification complexity that usually tries to cover all possible operational situations [e.g., a seven-port router has 49 (7^2) cases of communication]; and 4) it requires four different simulations for four different types of fault: soft error, hard fault, stuck-at-0, and stuck-at-1. When we increase the network size, the MTTF simulation time has significantly increased. On the other hand, the assessment time of the proposed method does not scale up with the network size. This is given by the assumption that the routers in the same position (corner, edge, side, or middle) inside the network have a similar fault rate. Therefore, the calculation can be reduced into (35) and (36). In fact, the obtained speedup with the proposed method is 791 209, 1 878 261, and 2 774 312 for $2 \times 2 \times 2$, $3 \times 3 \times 3$, and $4 \times 4 \times 4$ network sizes, respectively. The speedup values are expected to be much higher with larger network sizes.

In summary, the proposed analytical method provides an extremely fast solution to estimate the reliability of NoC systems. Although the proposed method is not as accurate as the MTTF simulation, its tremendous speedup values are very compelling for early system reliability assessment. In fact, to perform the MTTF simulation, we need to obtain a complete design and verification test, which may take several months of development. If the design cannot pass the reliability requirements, the waste in redesign time and resources can be extremely critical.

VII. CONCLUSION AND FUTURE WORK

In this paper, we proposed a reliability assessment for FT NoCs. The proposed method is based on three basic steps. First, the system is divided into subsystems. Second, a state model is built based on each subsystem and the reliability value can be obtained. Last, the final reliability of the system can be generated from its subsystems. In addition, we present an extended method for network's reliability that also helps designers.

Through extensive evaluations, we showed that the proposed method was acceptably matched with the simulation method while it reduces a large amount of modeling and simulation time and effort. This means that our method can provide a faster solution for FT systems. Before designing, researchers can apply the proposed method to estimate the enhancement in terms of reliability, which can help them understand the efficiency of the system.

In our assessment, we also point out the benefits of the FT system in terms of reliability improvement. Based on the pros and cons of the FT system, designers can select the appropriate mechanisms and configurations to match their requirements.

As a future work, we are planning to investigate the variation in fault rates, especially under the impacts of thermal variation and stress, to provide more accurate results.

ACKNOWLEDGMENT

The authors would like to thank all anonymous reviewers whose suggestions have helped to improve the quality of this paper.

REFERENCES

[1] A. B. Abderazek and M. Sowa, "Basic network-on-chip interconnection for future gigascale MCSoCs applications: Communication and computation orthogonalization," in *Proc. Symp. Sci., Soc., Technol. (TJASSST)*, 2006, pp. 1–7.

[2] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant NoCs," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, Apr. 2009, pp. 21–26.

[3] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge, "Reliability modeling and management in dynamic microprocessor-based systems," in *Proc. 43rd Annu. Design Autom. Conf.*, 2006, pp. 1057–1060.

[4] International Technology Roadmap for Semiconductors. (2015). "International Technology Roadmap for Semiconductors 2.0, section 5: More moore," International Technology Roadmap for Semiconductors, USA, Tech. Rep. 2015, accessed on Mar. 16, 2017. [Online]. Available: http://www.itrs2.net/itrs-reports.html

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

[6] K. Constantinides *et al.*, "Bulletproof: A defect-tolerant CMP switch architecture," in *Proc. IEEE 12th Int. Symp. High-Perform. Comput. Archit.*, Feb. 2006, pp. 5–16.

[7] X. Ni, E. Meneses, N. Jain, and L. V. Kalé, "Acr: Automatic checkpoint/restart for soft and hard error protection," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal. (ACM)*, 2013, pp. 7:1–7:12.

[8] C. Rowen, "Reducing SoC simulation and development time," *Computer*, vol. 35, no. 12, pp. 29–34, Dec. 2002.

[9] J. A. Cheatham, J. M. Emmert, and S. Baumgart, "A survey of fault tolerant methodologies for FPGAs," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 11, no. 2, pp. 501–533, 2006.

[10] A. Ickowicz, *Coordinating Committee, IEEE Guide for Selecting and Using Reliability Predictions*, IEEE Standards IEEE 1413.1, 2002.

[11] D. Crowe and A. Feinberg, *Design for Reliability*, vol. 11. Boca Raton, FL, USA: CRC Press, 2001.

[12] *United States of America: Department of Defense, Military Handbook: Reliability Prediction of Electronic Equipment*, document MIL-HDBK-217F, 1991.

[13] M. Zhang and N. R. Shanbhag, "Soft-error-rate-analysis (SERA) methodology," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2140–2155, Oct. 2006.

[14] A. Y. Yamamoto and C. Ababei, "Unified reliability estimation and management of NoC based chip multiprocessors," *Microprocess. Microsyst.*, vol. 38, no. 1, pp. 53–63, 2014.

[15] W. Najjar and J.-L. Gaudiot, "Network resilience: A measure of network fault tolerance," *IEEE Trans. Comput.*, vol. 39, no. 2, pp. 174–181, Feb. 1990.

[16] J. Chen, G. Wang, C. Lin, T. Wang, and G. Wang, "Probabilistic analysis on mesh network fault tolerance," *J. Parallel Distrib. Comput.*, vol. 67, no. 1, pp. 100–110, 2007.

[17] A. Eghbal, P. M. Yaghini, N. Bagherzadeh, and M. Khayambashi, "Analytical fault tolerance assessment and metrics for TSV-based 3D network-on-chip," *IEEE Trans. Comput.*, vol. 64, no. 12, pp. 3591–3604, Dec. 2015.

[18] A. Dalirsani, M. Hosseinabady, and Z. Navabi, "An analytical model for reliability evaluation of NoC architectures," in *Proc. 13th IEEE Int. On-Line Test. Symp. (IOLTS)*, Jul. 2007, pp. 49–56.

[19] S. Borkar, "Designing reliable systems from unreliable components: The challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, Nov. 2005.

[20] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," *IEEE Micro*, vol. 23, no. 4, pp. 14–19, Jul./Aug. 2003.

[21] T. Lehtonen, P. Liljeberg, and J. Plosila, "Online reconfigurable self-timed links for fault tolerant NoC," *VLSI Des.*, vol. 2007, Mar. 2007, Art. no. 94676, doi: 10.1155/2007/94676.

[22] 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 *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2012, pp. 60–71.

[23] R. Parikh and V. Bertacco, "Formally enhanced runtime verification to ensure NoC functional correctness," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2011, pp. 410–419.

[24] S. Shamshiri, A. Ghofrani, and K.-T. Cheng, "End-to-end error correction and online diagnosis for on-chip networks," in *Proc. IEEE Int. Test Conf. (ITC)*, Sep. 2011, pp. 1–10.

[25] H. T. Nguyen, Y. Yagil, N. Seifert, and M. Reitsma, "Chip-level soft error estimation method," *IEEE Trans. Device Mater. Rel.*, vol. 5, no. 3, pp. 365–381, Sep. 2005.

[26] A. Simevski, R. Kraemer, and M. Krstic, "Automated integration of fault injection into the ASIC design flow," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFTS)*, Oct. 2013, pp. 255–260.

[27] M. L. Shooman, *Reliability of Computer Systems and Networks: Fault Tolerance, Analysis, and Design*. Hoboken, NJ, USA: Wiley, 2003.

[28] J. B. Bernstein, M. Gurfinkel, X. Li, J. Walters, Y. Shapira, and M. Talmor, "Electronic circuit reliability modeling," *Microelectron. Rel.*, vol. 46, no. 12, pp. 1957–1979, 2006.

[29] K. Aisopos, C.-H. O. Chen, and L.-S. Peh, "Enabling system-level modeling of variation-induced faults in networks-on-chips," in *Proc. 48th Design Autom. Conf.*, 2011, pp. 930–935.

[30] D. Bertozzi, L. Benini, and G. D. Micheli, "Error control schemes for on-chip communication links: The energy-reliability tradeoff," *IEEE Trans. Comput.-Aided Des. Integr.*, vol. 24, no. 6, pp. 818–831, Jun. 2005.

[31] R. Cressent, P. David, V. Idasiak, and F. Kratz, "Designing the database for a reliability aware model-based system engineering process," *Rel. Eng. Syst. Safety*, vol. 111, pp. 171–182, Mar. 2013.

[32] J. A. Rivers, M. S. Gupta, J. Shin, P. N. Kudva, and P. Bose, "Error tolerance in server class processors," *IEEE Trans. Comput.-Aided Des. Integr.*, vol. 30, no. 7, pp. 945–959, Jul. 2011.

[33] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, and F. Gebalis, "A reliability-aware design methodology for networks-on-chip applications," in *Proc. IEEE 4th Int. Conf. Design Technol. Integr. Syst. Nanoscal Era (DTIS)*, Apr. 2009, pp. 107–112.

[34] L. Huang, F. Yuan, and Q. Xu, "Lifetime reliability-aware task allocation and scheduling for MPSoC platforms," in *Proc. Conf. Design, Autom. Test Eur.*, 2009, pp. 51–56.

[35] J. B. Bowles, "A survey of reliability-prediction procedures for microelectronic devices," *IEEE Trans. Rel.*, vol. 41, no. 1, pp. 2–12, Mar. 1992.

[36] J. Shin, V. Zyuban, Z. Hu, J. A. Rivers, and P. Bose, "A framework for architecture-level lifetime reliability modeling," in *Proc. 37th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2007, pp. 534–543.

[37] T. Lehtonen, P. Liljeberg, and J. Plosila, "Fault tolerance analysis of NoC architectures," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 361–364.

[38] H. Zimmer and A. Jantsch, "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip," in *Proc. 1st IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, Oct. 2003, pp. 188–193.

[39] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, and C. R. Das, "Design and analysis of an NoC architecture from performance, reliability and energy perspective," in *Proc. ACM Symp. Archit. Netw. Commun. Syst.*, 2005, pp. 173–182.

[40] A. Savino, S. Di Carlo, G. Politano, A. Benso, A. Bosio, and G. Di Natale, "Statistical reliability estimation of microprocessor-based systems," *IEEE Trans. Comput.*, vol. 61, no. 11, pp. 1521–1534, Nov. 2012.

[41] M. Khayambashi, P. M. Yaghini, A. Eghbal, and N. Bagherzadeh, "Analytical reliability analysis of 3D NoC Under TSV failure," *J. Emerg. Technol. Comput. Syst.*, vol. 11, no. 4, pp. 43:1–43:16, Apr. 2015.

[42] A. Bobbio and K. S. Trivedi, "An aggregation technique for the transient analysis of stiff Markov chains," *IEEE Trans. Comput.*, vol. C-35, no. 9, pp. 803–814, Sep. 1986.

[43] K. N. Dang, M. Meyer, Y. Okuyama, and A. B. Abdallah, "Reliability assessment and quantitative evaluation of soft-error resilient 3D network-on-chip systems," in *Proc. IEEE 25th Asian Test Symp.*, Nov. 2016, pp. 161–166.

[44] J. P. Hayes, "A graph model for fault-tolerant computing systems," *IEEE Trans. Comput.*, vol. C-100, no. 9, pp. 875–884, Sep. 1976.

[45] K. N. Dang, M. Meyer, Y. Okuyama, and A. B. Abdallah, "A low-overhead soft–hard fault-tolerant architecture, design and management scheme for reliable high-performance many-core 3D-NoC systems," *J. Supercomput.*, vol. 73, no. 6, pp. 2705–2729, 2017.

[46] A. B. Ahmed and A. B. Abdallah, "LA-XYZ: Low latency, high throughput look-ahead routing algorithm for 3D network-on-chip (3D-NoC) architecture," in *Proc. IEEE 6th Int. Symp. Embedded Multicore Socs (MCSoC)*, Sep. 2012, pp. 167–174.

[47] A. B. Ahmed and A. B. Abdallah, "Adaptive fault-tolerant architecture and routing algorithm for reliable many-core 3D-NoC systems," *J. Parallel Distrib. Comput.*, vols. 93–94, pp. 30–43, Jul. 2016.

[48] K. N. Dang, M. Meyer, Y. Okuyama, X.-T. Tran, and A. B. Abdallah, "A soft-error resilient 3D network-on-chip router," in *Proc. IEEE 7th Int. Conf. Awareness Sci. Technol. (iCAST)*, Sep. 2015, pp. 84–90.

[49] NCSU Electronic Design Automation. *FreePDK3D45 3D-IC Process Design Kit*. accessed on Jun. 16, 2016. [Online]. Available: http://www.eda.ncsu.edu/wiki/FreePDK3D45:Contents

[50] NanGate Inc., *Nangate Open Cell Library 45 nm*. accessed on Jun. 16, 2016. [Online]. Available: http://www.nangate.com/

[51] Q. Yu, M. Zhang, and P. Ampadu, "Addressing network-on-chip router transient errors with inherent information redundancy," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 4, pp. 105:1–105:21, 2013.

[52] G. Van Rossum, "An introduction to Python for UNIX/C programmers," in *Proc. NLUUG Najaarsconf. Dutch UNIX Users Group*, 1993, pp. 1–8.

Khanh N. Dang received the B.Sc. degree in electronics telecommunication technology from the VNU University of Engineering and Technology, Hanoi, Vietnam, in 2011, and the M.Sc. degree in information, systems, and technology from the University of Paris-Sud XI, Orsay, France, in 2014. He is currently pursuing the Ph.D. degree with The University of Aizu, Aizuwakamatsu, Japan.

His current research interests include system-on-chips/network-on-chips, 3-D-ICs, and fault-tolerant systems.

Akram Ben Ahmed received the M.S.E. and Ph.D. degrees in computer science and engineering from The University of Aizu, Aizuwakamatsu, Japan, in 2012 and 2015, respectively.

He is currently a Post-Doctoral Researcher with the Department of Information and Computer Science, Keio University, Yokohama, Japan. His current research interests include on-chip interconnection networks, reliable and fault-tolerant systems, and ultralow-power embedded real-time systems.

Xuan-Tu Tran received the Ph.D. degree in micro and nanoelectronics from Grenoble INP (at the CEA-LETI), Grenoble, France, in 2008.

He was an Invited Professor with the University of Paris-Sud XI, Orsay, France, in 2009, 2010, and 2015, and a Visiting Professor with Grenoble INP in 2011. He is currently an Associate Professor with the VNU University of Engineering and Technology, Hanoi, Vietnam, where he is also with the Key Laboratory for Smart Integrated Systems.

Yuichi Okuyama received the master's and Ph.D. degrees in computer science and engineering from The University of Aizu, Aizuwakamatsu, Japan, in 1999 and 2002, respectively.

He is currently an Associate Professor with The University of Aizu. His current research interests include reconfigurable architecture design, parallel programming for pattern recognition, and education of computer fundamentals.

Abderazek Ben Abdallah received the Ph.D. degree in computer engineering from the University of Electro-Communications at Tokyo, Chofu, Japan, in 2002.

He was a Research Associate with the Graduate School of Information Systems, University of Electro-Communications at Tokyo, from 2002 to 2007. He is currently a Full Professor of computer science and engineering and the Head of the Division of Computer Engineering, The University of Aizu, Aizuwakamatsu, Japan. He has been a Faculty Member with The University of Aizu since 2007. He has authored three books, published over 150 journal articles and conference papers in these areas, and given invited talks and courses at several universities. His current research interests include the area of computer system and architecture, with an emphasis on adaptive/self-organizing systems, networks-on-chip/system-on-chips, processor microarchitecture, power and reliability-aware architectures, neuroinspired systems, and VLSI design for 3-D-ICs.

Dr. Ben Abdallah is a Senior Member of ACM and a member of IEICE. He has been a Principal Investigator or a Co-Principal Investigator of several projects for developing next-generation high-performance reliable computing systems for applications in general purpose and pervasive computing.