Fault Tolerance and Reliability for Partially Connected 3D Networks-on-Chip

Alexandre Augusto da Penha Coelho

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Nano Électronique & Nano Technologies**

Arrêté ministériel : 25 mai 2016

Présentée par

Alexandre Augusto da PENHA COELHO

Thèse dirigée par **Raoul VELAZCO**
et codirigée par **Nacer-Eddine ZERGAINOH**

préparée au sein du **Laboratoire Techniques de l'Informatique et de la Microélectronique pour l'Architecture des systèmes intégrés**
dans **l'École Doctorale Electronique, Electrotechnique, Automatique, Traitement du Signal (EEATS)**

Tolérance aux fautes et fiabilité pour les réseaux sur puce 3D partiellement connectés

Fault Tolerance and Reliability for Partially Connected 3D Networks-on-Chip

Thèse soutenue publiquement le **25 Octobre 2019**,
devant le jury composé de :

Monsieur Raoul VELAZCO
Directeur de Recherche, CNRS Délégation Alpes , Directeur de thèse
Monsieur Nacer-Eddine ZERGAINOH
Maître de Conférences, Université Grenoble Alpes, Co-Directeur de thèse
Madame Lirida NAVINER
Professeur, Telecom ParisTech, Président du jury
Monsieur Gilles SASSATELLI
Directeur de Recherche, CNRS Délégation Occitanie Est, Rapporteur
Monsieur Amer BAGHDADI
Professeur, IMT Atlantique Bretagne-Pays de la Loire, Rapporteur

Dear Lord, thank you for giving me the strength and the conviction to complete the task you entrusted to me. Thank you for guiding me straight and true through the many obstacles in my path. And for keeping me resolute when all around seemed lost. Thank you for your protection and your many signs along the way. Thank you for the friends I made during my PhD. And thank you for finally allowing me to finish this thesis, I am so very tired. I fought the good fight, I finished the race, I kept the faith.

- Adapted from "Book of Eli Movie"

To Cristina, Aline, Linus and Alexia

Acknowledgement

I would like to express my sincere gratitude to my advisors Dr. Raoul Velazco and Dr. Nacer-Eddine Zergainoh, who gave me the chance to work with them and guided me through this harsh road. Their guidance undoubtedly helped me to improve my research and mitigate all the insecurities a Ph.D. student may have. I want to express my special thanks to Prof. Lirida Naviner, Dr. Gilles Sassatelli, and Prof. Amer Baghdadi, who accepted being part of my examining board. Their valuable and constructive feedback went a long way towards improving the quality of this manuscript and the dissertation defense.

I want(would like to) to thank the Brazilian National Council for Scientific and Technological Development (CNPq Brazil) for their financial support.

I would like to thank all the staff of TIMA Laboratory, EEATS, and CIME Nanotech, who somehow always managed to help me with countless issues. I would like to name a few friends whose collaboration impacted decisively in the result of this work: Amir Charif, Juan Fraire, Miguel Solinas, Matheus Garay, Rodrigo Possamai, Thiago Leite, Renato Feitoza, David Saraiva, and Jasmina Karajanov (italki). Their feedbacks and friendship were essential for the final result of this work.

I would like to thank some people that were important in my decision to do a Ph.D.: Sobral, Helano, Cortez, Jardel, and Jarbas. Thank you guys very much for supporting and encouraging me during this arduous journey. (arduous journey)

I would like to extend great thanks to my dear Vozinha, to whom I dedicate this thesis. I am grateful for her teaching, love, kindness, and affection. I also express my gratitude to my mother-in-law Lais. Her passion, love, and dedication to taking care of my children are incredible. Without her support, I could not finish this thesis.

I would like to thank my loved ones, from whom I had to stay apart since I decided to move to France. My sisters Karine, Isabel, Rebeca, and my niece and nephews Ana Mel, Carlos, Arthur, and João Gabriel. Furthermore, I would like to thank my Dad for introducing me to the road of computers and electronics at an early age, and also for teaching me to think critically.

I would like to thank my beloved wife, Aline, for loving me as I am, for all these happy years of marriage, and her support and inspiration during the development of this thesis. Thank you again for your motivation, encouragement, and patience. I love you so much. Also, I want to thank my children, Linus and Alexia, for being my inspiration, for their love, tenderness, and

for their smiles that have decorated my life.

Finally, I would especially like to thank my Mother for everything she did for me. She was the best Mother I could ever have imagined having. This work could not have been done without her love. Thanks for teaching me to dream and for always believing in me, even when I do not believe in myself. Mother, you are eternally my inspiration. I miss you so much. I would change everything to spend more time with you...

Table of Contents

List of Figures

List of Tables

Abstract

Networks-on-Chip (NoC) have emerged as a viable solution for the communication challenges in highly complex systems-on-chip. The NoC architecture paradigm, based on a modular packet-switched mechanism, can address many of the on-chip communication challenges such as wiring complexity, communication latency, and bandwidth. Furthermore, the combined benefits of 3D IC and NoC schemes provide the possibility of designing a high-performance system in a limited chip area. The major advantages of Three-Dimensional Networks-on-Chip (3D-NoCs) are a considerable reduction in the average wire length and wire delay, resulting in lower power consumption and higher performance. However, 3D-NoCs suffer from some reliability issues such as the process variability of 3D-IC manufacturing. In particular, the low yield of vertical connection significantly impacts the design of three-dimensional die stacks with a large number of Through Silicon Vias. Equally concerning, advances in integrated circuit manufacturing technologies are resulting in a potential increase in their sensitivity to the effects of radiation present in the environment in which they will operate. In the past, this issue was exclusively related to space applications, while nowadays it must be taken into account for any application operating in the Earth's atmosphere whose errors can have critical consequences. In fact, the increasing number of transient faults has become, in recent years, a major concern in the design of critical System-on-Chip. As a result, the evaluation of the sensitivity of circuits and applications to events caused by energetic particles present in the real environment is a major concern that needs to be addressed. So, this thesis presents contributions in two important areas of reliability research:

- In the design and implementation of deadlock-free fault-tolerant routing schemes for the emerging three-dimensional Networks-on-Chips.
- In the design of fault injection frameworks able to emulate single and multiple transient faults in the HDL-based circuits.

The first part of this thesis addresses the issues of transient and permanent faults in the architecture of 3D-NoCs and introduces a new resilient routing computation unit as well as a new runtime fault-tolerant routing scheme. A novel resilient mechanism is introduced in order to tolerate transient faults occurring in the route computation unit (RCU), which is the most important logical element in NoC routers. Failures in the RCU can provoke misrouting, which may lead to

severe effects such as deadlocks or packet loss, corrupting the operation of the entire chip. By combining a reliable fault detection circuit leveraging circuit-level double-sampling, with a cost-effective rerouting mechanism, we develop a full fault-tolerance solution that can efficiently detect and correct such fatal errors before the affected packets leave the router. To validate the proposed solution, we also have introduced a novel method for simulation-based fault-injection based on NoC's gate-level netlist. Experimental results obtained from a vertically-partially-connected 3D Network-on-Chip indicate that our solution can provide a high level of reliability in the presence of errors, at the expense of low area and power overhead.

The first part of this thesis also describes a novel fault-tolerant routing scheme for vertically-partially-connected 3D Networks-on-Chip called FL-RuNS, constructed using our previous routing algorithm First-Last as baseline. Thanks to an asymmetric distribution of virtual channels, our fault-tolerant routing scheme can guarantee 100% packet delivery under an unconstrained set of runtime and permanent vertical link failures. This scheme requires a very low number of asymmetric virtual channels to achieve booth deadlock-freedom and reliability. Also, FL-RuNS uses a runtime mechanism to dynamically and progressively reconfigure the network without any packet loss. Simulation results demonstrate the effectiveness of our approach in terms of performance and reliability when compared with the state-of-the-art routing algorithm. Furthermore, the hardware synthesis performed using a commercial 28nm technology library shows a reasonable area and power overhead with respect to the non-fault-tolerant baseline.

With the aim to emulate the radiation effects on new systems-on-chip designs, the second part of this thesis addresses the fault injection methodologies by introducing two frameworks named NETFI-2 (Netlist Fault Injection) and NoCFI (Networks-on-Chip Fault Injection). NETFI-2 is a fault injection methodology able to emulate transient faults such as Single Event Upsets (SEU) and Single Event Transient (SET) in a HDL-based (Hardware Description Language) design. NETFI-2 was constructed using as baseline an existing fault injection framework developed in the TIMA Laboratory. NETFI-2 allows injecting SEUs and SETs from a single FPGA without external controllers, while allowing to choose the combinational logic granularity to better emulate SETs. Extensive experiments performed on two appealing case studies are presented to demonstrate NETFI-2 features and advantages. Finally, in the last part of this work, we present NoCFI as a novel methodology to inject multiple faults such as MBUs and SEMT in a Networks-on-Chip architecture. NoCFI combines ASIC-design-flow, in order to extract layout information, and FPGA-design-flow to emulate multiple transient faults. In order to validate the NoCFI's methodology, a two-dimensional NoC was used as a study case.

Keywords: Three-dimensional Networks-on-Chip, Soft-errors, Fault-tolerant routing algorithm, transient and permanent faults

Résumé

Les réseaux sur puce (NoC) sont apparus comme une solution viable aux problèmes de communication dans les systèmes sur puce très complexes (SoC). Le paradigme de l'architecture NoC, basé sur un mécanisme modulaire de commutation par paquets, peut répondre à de nombreux défis de communication sur puce tels que la complexité du câblage, la latence des communications et la bande passante. De plus, les avantages combinés des schémas 3D IC et NoC offrent la possibilité de concevoir un système haute performance dans une zone de puce limitée. Les principaux avantages des réseaux tridimensionnels sur puce (3D-NoCs) sont une réduction considérable de la longueur moyenne des fils et du temps de propagation des fils, ce qui se traduit par une consommation d'énergie moindre et des performances supérieures. Cependant, les NoCs 3D souffrent de certains problèmes de fiabilité tels que la variabilité des processus de fabrication 3D-IC. En particulier, le faible rendement de la connexion verticale a un impact significatif sur la conception des piles de matrices tridimensionnelles avec un grand nombre de trous traversants en silicium. De même, les progrès des technologies de fabrication de circuits intégrés entraînent une augmentation potentielle de leur sensibilité aux effets des rayonnements présents dans l'environnement dans lequel ils vont fonctionner. Dans le passé, cette question était exclusivement liée aux applications spatiales, alors qu'aujourd'hui elle doit être prise en compte pour toute application opérant dans l'atmosphère terrestre dont les erreurs peuvent avoir des conséquences critiques. En fait, le nombre croissant de défaillances transitoires est devenu, ces dernières années, une préoccupation majeure dans la conception des systèmes sur puce critiques. Par conséquent, l'évaluation de la sensibilité des circuits et des applications aux événements causés par les particules énergétiques présentes dans l'environnement réel est une préoccupation majeure à laquelle il faut répondre. Cette thèse présente donc des contributions dans deux domaines importants de la recherche sur la fiabilité :

- Dans la conception et la mise en œuvre de schémas de routage à tolérance de pannes sans blocage pour les réseaux sur puce tridimensionnels émergents.
- Dans la conception de cadres d'injection de fautes capables d'émuler des fautes transitoires simples et multiples dans les circuits à base de HDL.

La première partie de cette thèse aborde les problèmes des défauts transitoires et permanents dans l'architecture des NoCs 3D et présente une nouvelle unité de calcul de routage résiliente

ainsi qu'un nouveau schéma de routage tolérant aux défauts d'exécution. Un nouveau mécanisme résilient est introduit afin de tolérer les défauts transitoires se produisant dans l'unité de calcul de route (RCU), qui est l'élément logique le plus important dans les routeurs NoC. Les défaillances de la télécommande peuvent provoquer des erreurs d'acheminement, ce qui peut entraîner des effets graves tels que des blocages ou la perte de paquets, corrompant le fonctionnement de la puce entière. En combinant un circuit de détection de défauts fiable à double échantillonnage au niveau du circuit et un mécanisme de réacheminement économique, nous développons une solution complète de tolérance aux fautes qui peut détecter et corriger efficacement ces erreurs fatales avant que les paquets affectés ne quittent le routeur. Pour valider la solution proposée, nous avons également introduit une nouvelle méthode d'injection de défaillances basée sur la simulation basée sur la liste de réseau au niveau de la porte de NoC. Les résultats expérimentaux obtenus à partir d'un réseau sur puce 3D à connexion verticale partielle indiquent que notre solution peut fournir un haut niveau de fiabilité en présence d'erreurs, au détriment d'une faible surface et d'une surcharge électrique.

La première partie de cette thèse décrit également un nouveau schéma de routage tolérant aux pannes pour les réseaux 3D connectés verticalement sur puce, appelé FL-RuNS, construit en utilisant notre algorithme de routage précédent First-Last comme référence. Grâce à une distribution asymétrique des canaux virtuels, notre système de routage tolérant aux pannes peut garantir une livraison de paquets à 100% dans le cadre d'un ensemble illimité de temps d'exécution et de pannes permanentes des liaisons verticales. Ce système nécessite un très faible nombre de canaux virtuels asymétriques pour garantir la liberté et la fiabilité des cabines dans l'impasse. De plus, FL-RuNS utilise un mécanisme d'exécution pour reconfigurer dynamiquement et progressivement le réseau sans perte de paquets. Les résultats de la simulation démontrent l'efficacité de notre approche en termes de performance et de fiabilité par rapport à l'algorithme de routage le plus moderne. De plus, la synthèse matérielle effectuée à l'aide d'une bibliothèque technologique commerciale de 28nm montre une surface et une surcharge électrique raisonnables par rapport à la référence non tolérante aux pannes.

Dans le but d'émuler les effets du rayonnement sur les nouvelles conceptions de systèmes sur puce, la deuxième partie de cette thèse aborde les méthodologies d'injection de fautes en introduisant deux frameworks nommés NETFI-2 (Netlist Fault Injection) et NoCFI (Networks-on-Chip Fault Injection). NETFI-2 est une méthodologie d'injection de fautes capable d'émuler les fautes transitoires telles que les perturbations d'événement unique (Single Event Upsets - SEU) et les transitoires d'événement unique (Single Event Transient - SET) dans une conception basée sur HDL (Hardware Description Language). NETFI-2 a été construit en utilisant comme référence un cadre d'injection de fautes existant développé dans le laboratoire TIMA. NETFI-2 permet d'injecter des SEU et des SETs à partir d'un seul FPGA sans contrôleurs externes, tout en permettant de choisir la granularité logique combinatoire pour mieux émuler les SETs. Des expériences approfondies réalisées sur deux études de cas attrayantes sont présen-

tées pour démontrer les caractéristiques et les avantages de la NETFI-2. Enfin, dans la dernière partie de ce travail, nous présentons NoCFI comme une nouvelle méthodologie pour injecter des défauts multiples tels que les MBU et SEMT dans une architecture de réseaux sur puce. NoCFI combine ASIC-design-flow, afin d'extraire les informations de layout, et FPGA-design-flow pour émuler plusieurs défauts transitoires. Afin de valider la méthodologie du NoCFI, un NoC bidimensionnel a été utilisé comme cas d'étude.

Keywords: Réseaux tridimensionnels sur puce, erreurs logicielles, algorithme de routage tolérant aux pannes, pannes transitoires et permanentes

Part I

INTRODUCTION

Chapter 1

Introduction

The continuous advances in semiconductor technologies make possible to integrate billions of gates into a single chip [11]. The availability of such abundant resources have enabled designers to fabricate chips with tens or hundreds of Processing Elements (PE) blocks on a single chip, resulting in the conception of Multiprocessor System-on-Chips (MPSoC). For a MPSoCs on such massive scale, connectivity is a major concern, and inefficient/unreliable interconnects can severely limit performance. For example, MPSoCs architectures with two or four cores typically use the traditional on-chip bus communication between cores. However, as the number of cores on a chip increases, the communication between the cores plays a crucial role in its performance. Due to the bus's lack of scalability, Networks-on-Chip (NoCs) design has emerged as a scalable on-chip interconnection network that can efficiently handle the strict communication requirements between cores on a chip. It has happened thanks to the ability of NoCs supporting simultaneous communication between multiple pairs of cores.

On the other hand, while the same technological evolution to nanometric scaling process has its benefits in terms of delay, area, and power consumption, it is known to pose some serious reliability concerns. It means that circuits in nanoscale era are more sensitive to failures caused at the manufacturing process, due to process variations, or even simply being exposed to harsh environments, such as space. Despite the sensitivity and some reliability concerns, a careful fault-tolerant design still promises to achieve reliable systems from both MPSoCs and NoCs architectures. And, of course, the development of a new fault-tolerant design calls for evaluation and validation of these mechanisms at an early design stage. In other words, to cope with permanent and transient faults, designers need to be able to evaluate their impact, implement suitable error mitigation techniques, and validate the results.

However, the analysis of permanent and transient faults in the MPSoCs and NoCs architectures is not as trivial as it may seem. It is due to the fact that these failures must be investigated when the circuit is in operation (i.e., on-the-fly) which may lead the circuit to an unpredictable behavior. Also, some failures caused by multiple-cell upset (MCU), multiple-bit upsets (MBU), and single-event multiple transient (SEMT) that were not considered in the older technology,

due to its inherent ability to mask these failures, are now a major concern that needs to be addressed in the new MPSoCs and NoCs architectures.

One of the most popular techniques to evaluate the reliability of digital circuits is to submit those circuits to a radiation test campaign under a large particle flow [31]. However, these campaigns are very expensive and are usually performed on the final physical implementation of the circuit. Consequently, the sensitive parts of the circuits which require mitigation can only be detected after the radiation campaign. It may result in re-working project phases, in new manufacturing processes, and in new radiation campaigns that increase the budget of the project as well as its time-to-market. So, a solution for this problem is to verify if the circuits satisfy fault tolerance requirements during its project phases instead of only in its final prototype. With this in mind, *Simulation-based* and *Emulated-based* fault injection are two widely adopted methods to analyze the effects of transient faults in (Hardware Description Language) HDL-based design [94] already in its initial project phases. Also, new tools to test and validate multiple bits upset taking adjacent cell into account are necessary to better understand the behavior of the circuit under permanent and transient faults.

Emerging technology such as Systems-on-Chip design based on machine learning is being increasingly implemented in integrated circuits instead of only in a software application. Also, MLSoCs (Machine Learning Systems-on-Chip) have been employed in critical applications due to their capacity to predict errors and learn from their own decisions. Those two features, learning and predicting, have motivated the use of this type of algorithm in many other applications such as medical diagnostics [7], robot intelligence [53], and geoscience/aerospace domain [62]. But, MLSoCs suffer from the same reliability issues as all standard digital circuits, which means that they also must be evaluated in the context of reliability.

In summary, the study of new fault-tolerant solutions for both SoCs and NoCs is mandatory in the nanoscale era. Also, new high-accuracy fault injection approaches able to evaluate and validate the effects of permanent and transient faults in the future SoC/NoC design are essential. The reliability issues presented above motivated us to propose in this thesis contributions in two crucial areas of microelectronic research:

- In design and implementation of deadlock-free fault-tolerant routing scheme for the emerging three-dimensional Networks-on-Chips.

- In design of fault injection frameworks able to emulate single and multiple transient faults in the HDL-based circuits.

All contributions of this thesis are summarized in the remainder of this Section.

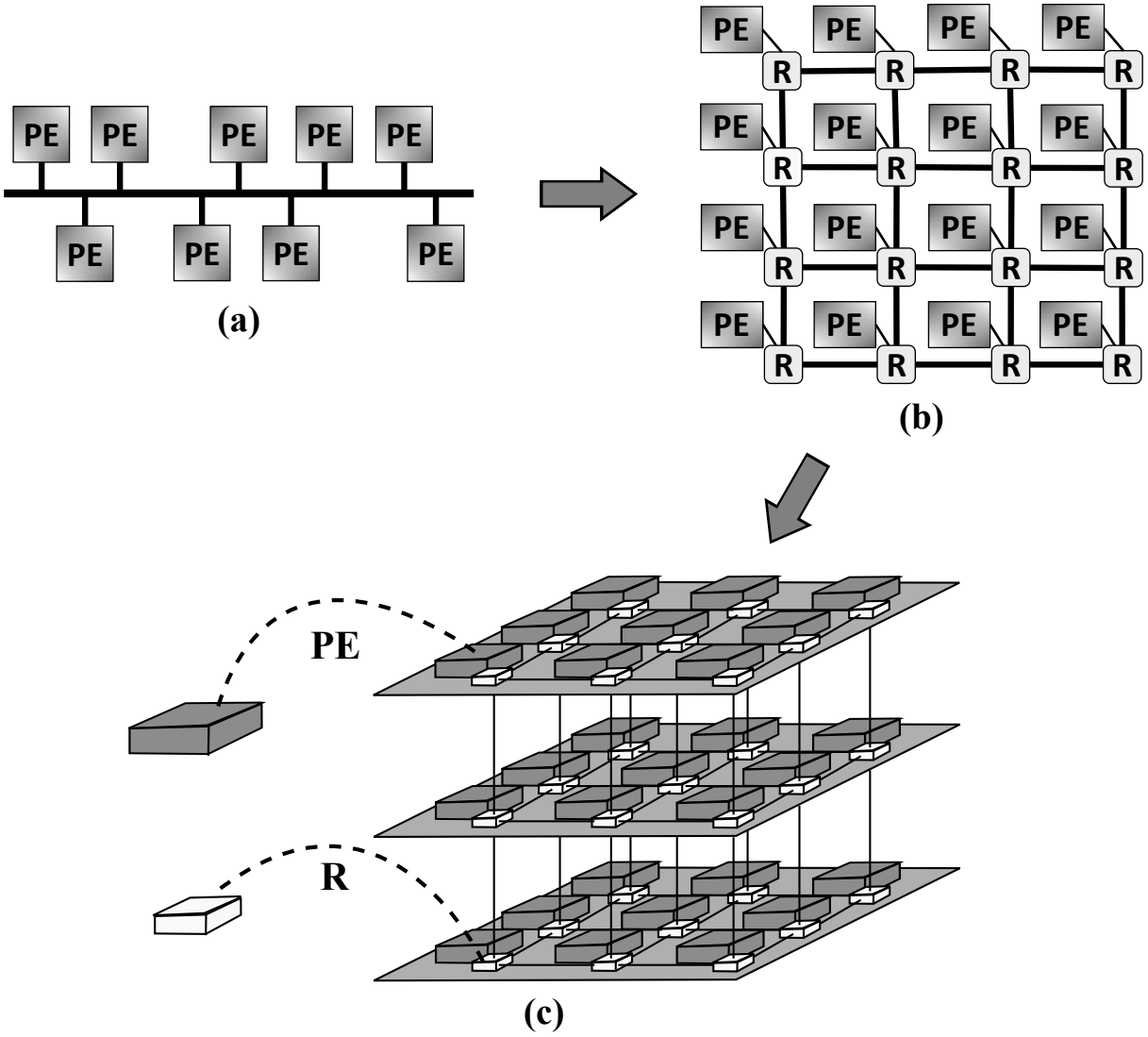

Fig. 1.1: Evolution of the System-On-Chip intra-chip communication architecture from (a) the bus architecture to (b) the two-dimensional Networks-on-Chip, and finally to (c) the three-dimensional Networks-on-Chip.

1.1 Contribution I: Fault-Tolerant Solutions for 3D Networks-on-Chips

Traditionally, System-on-Chip (SoC) designers employ buses or hierarchical bus structures to interconnect Processing Elements (PE) blocks [42]. However, as chip integration grows, the global bus-based interconnection has become a bottleneck for future high-performance SoC designs [33]. To overcome this limitation, Networks-on-Chip has emerged as a promising infrastructure for on-chip communication due to its scalability, high bandwidth, better throughput, and lower power consumption [98]. However, wire delay and power consumption increase significantly by the usage of global interconnections in Two-Dimensional Integrated Circuit (2D-IC) designs. In other words, the restricted floor-planning choices of 2D-IC designs limit the

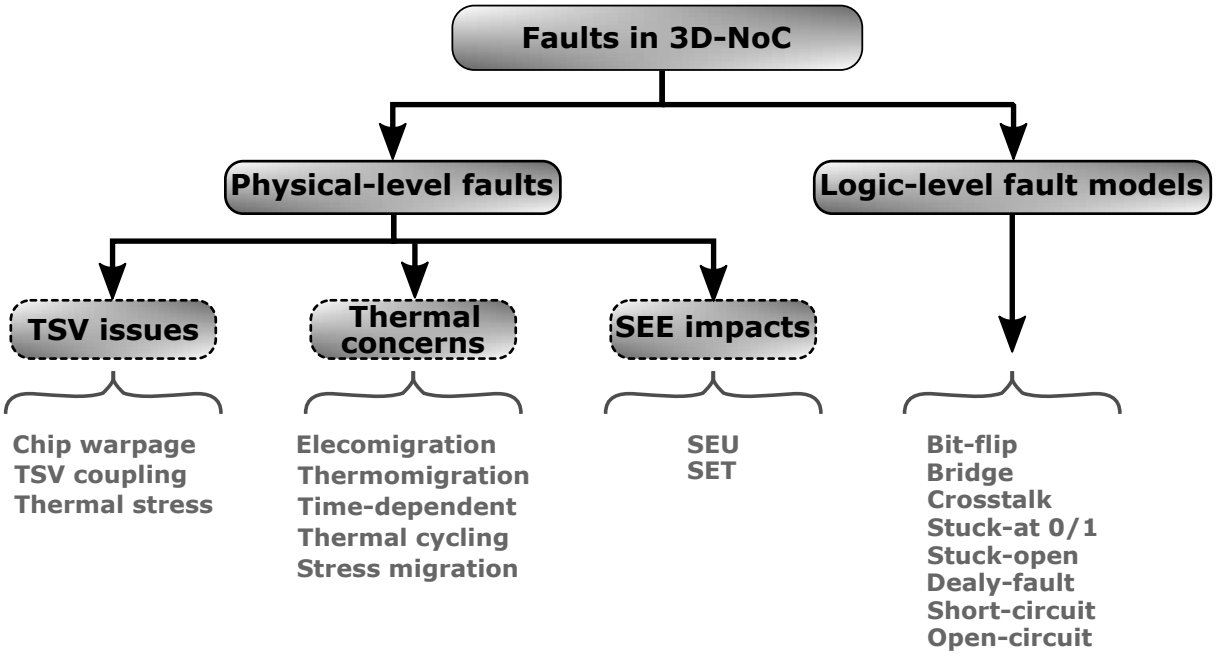

Fig. 1.2: Types of failures in a three-dimensional Networks-on-Chip architectures from [45].

potential performance of two-dimensional Networks-on-Chip (2D-NoC) architecture [50, 78]. In order to extend 2D-IC capabilities, multiple layers of active devices are integrated using vertical high-speed interconnection in a three-dimensional integrated circuit (3D-IC) architecture [35]. This approach permits the use of three-dimensional Networks-on-Chip (3D-NoC) as a communication infrastructure that reduces the interconnection lengths and improves the overall performance. Figure 1.1 shows the evolution of the on-chip interconnection from bus-based to three-dimensional Networks-on-Chip.

The need for reliability assessment in order to figure out the underlying process variation of NoC architecture has become more critical as CMOS technology continues to shrink. Figure 1.2 summarizes the potential faults, affecting the performance of 3D-NoC design and divides them into subcategories [45]. In this thesis, our contributions for 3D-NoC architecture are mainly focused on the fault-tolerant solutions to mitigate the **SEE impacts** and **TSV issues**. The first part of this thesis focuses on:

- Investigating the effects of transient faults in the sequential and combinational logic of 3D-NoC's routers. New fault-tolerant solutions that can mitigate these effects of transient faults in the control logic of the 3D-NoC's routers are introduced.
- Improving the reliability of the 3D-NoCs when transient and/or permanent failures occur in the vertical links. To this end, we propose online-fault-tolerant schemes that permit the continued operation of the network in the presence of failures in booth manufacture and execution time.
- Proposing fault-tolerant techniques for 3D-NoCs architecture that can be feasible in terms

of hardware employing the ASIC design flow. It means that all solutions and mechanisms introduced throughout this thesis are implemented and synthesized. Also, in order to make a fair comparison in terms of hardware overhead, all our contributions, as well as the existent state-of-the-art architectures, were synthesized using the 28nm design kit from ST-Microelectronics.

In this context, in Chapter 2, we introduce some strategies to tolerate soft-errors in the Routing Computation Unit (RCU). Then in Chapter 3, we propose a runtime and reconfigurable routing scheme able to tolerate transient and permanent faults in the vertical connections of the 3D-NoC. It is worth mentioning that the proposed solutions avoid dropping packets while the 3D-NoC is being recovered from failures.

1.1.1 Strategies to deal with soft-errors in 3D-NoCs

The increasing complexity of 3D-NoC routers, the continuous miniaturization of silicon technology, the lower operating voltages, and the higher operating frequencies have made the 3D-NoC increasingly vulnerable to soft errors. In particular, transient faults occurring in the route computation unit (RCU) can provoke misrouting, as packets may be directed to the wrong output port. This may lead to severe effects such as deadlocks or packet loss that can corrupt the operation of the entire chip. It is therefore mandatory to provide some level of protection against routing errors. So, in Chapter 2, we have proposed some fault-tolerant solutions to mitigate the effects of transient faults in the RCU. In other words, a full fault-tolerance solution that can efficiently **detect** and then **correct** fatal errors in the RCU before the affected packets leave the router is described and validated. The main contributions of Chapter 2 are summarized below:

- We have introduced a reliable fault detection circuit which works in parallel with pipeline of the control path of the NoCs without incurring any path delay.
- We have proposed a fault detection strategy based on the concept of double-sampling.
- We have proposed a solution for recovering from faults based on a cost-effective rerouting mechanism. In other words, when the failures provoke errors that cannot be masked by the double-sampling techniques, the RCU must recompute the packet to avoid deadlock as well as packet loss.
- we have validated our fault-tolerant solutions by means of a novel simulation-based fault-injection methodology which is based on the NoC's gate-level netlist.

To provide the reader with a better understanding of the on-chip interconnection networks, a brief overview of three-dimensional networks-on-Chip architecture is described in Section 2.3

Some parts of this work was published in [25].

1.1.2 The FL-RuNS Fault-Tolerant Routing Scheme

Since three-dimensional Networks-on-Chip (3D-NoC) have been accepted as an effective solution to the scalability and latency issues in modern complex System-On-Chips, Through-Silicon Via (TSV) has been usually adopted as a viable technology enabling vertical connection among NoC layers. However, TSV-based architectures typically exhibit high vulnerability to transient and permanent faults, calling for robust routing solutions capable of sustaining operation under unpredictable failure patterns. Those issues motivated us to propose a complete fault-tolerant routing scheme named FL-RuNS that guarantees 100% packet delivery under an unconstrained set of runtime and permanent vertical link failures. So, in Chapter 3, we explore, for the first time, a fault-tolerant scheme for assigning TSVs to routers both offline and during runtime phase. The contributions in this domain can be summarized as follows:

- We have introduced a fault-tolerant scheme that requires a very low number of virtual channels to achieve booth deadlock-freedom and reliability. Our methodology is based on an asymmetric use of virtual channels instead of the usual symmetry approach. The idea is to provide a tradeoff among a fault-tolerant scheme, hardware overhead, and high throughput.
- we have proposed a runtime mechanism to dynamically and progressively reconfigure the network without any packet loss. It means that when faults are detected in the vertical connection, the network can be reconfigured online without stopping the operation of the network and/or dropping packets.
- we have validated the effectiveness of our approach in terms of performance and reliability using simulation and comparing its results with the state-of-the-art routing algorithm. Also, the hardware synthesis performed on our approach shows a reasonable area and power overhead compared to the non-fault-tolerant baseline.

A preliminary version of the FL-RuNS routing scheme was published in [28] and [24]

1.2 Contribution II: Automated Fault Injection Tools for HDL Based Design

Radiation on integrated circuits can cause a wide variety of effects. As shown in Figure 1.3, a single energetic particle (neutron, proton, heavy ion or alpha particle) when interacting with the semiconductor material can produce a destructive or non-destructive event. Among them, nonpermanent single-event effects (SEEs), also known as soft-errors, have the potential for inducing the highest failure rate of all other reliability mechanism combined [40, 56, 71]. The most importants SEEs of this type are single-event upset (SEU) and the single-event transient

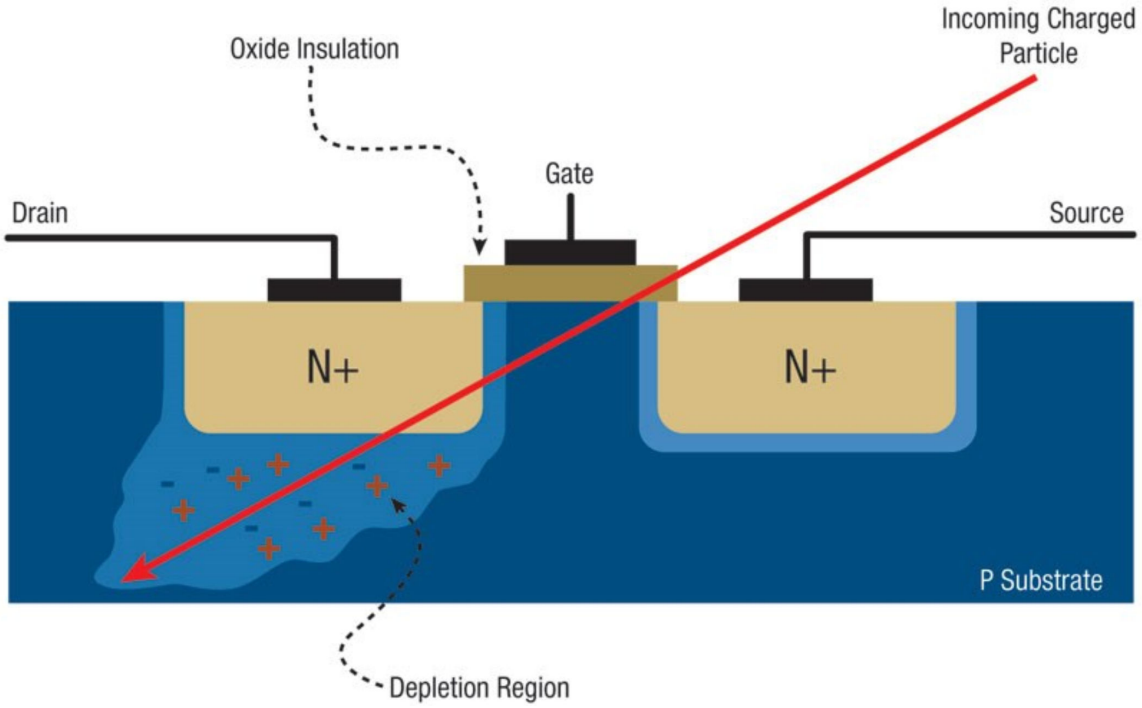

Fig. 1.3: Radiation strike causing transistor disruption.

(SET). So, radiation environment and their effects must be taken into account early enough in the design cycle for any application devoted to operating in the Earth's atmosphere of which a fault may have critical consequences. As a result, the demand for new tools able to test emerging circuits already in their project phase has been growing in the last years. In particular, there has been an increasing demand for tools that can emulate SEUs and SETs with high accuracy in the circuits under test (CUT) with a minimum modification in the CUT's structure.

Fault injection is a well-known technique to evaluate the sensitivity of integrated circuits to the effects of radiation. For this reason, fault injection has been a research topic for decades [94, 95] and has been classified in two main categories: Simulation-based and Emulation-based fault injection methodologies. Simulation-based approaches use a digital simulator to execute HDL-based design and perform fault injections models. On the other hand, emulation-based approaches are normally based on FPGAs, where both the emulation of the circuit under test (CUT) as well as the fault injection campaign are performed.

In the second part of this thesis, we propose two fault injection frameworks to study the effects of transient faults in the HDL-based circuits. In the first framework presented in Chapter 4, an existing emulation-based methodology called NETFI [73] is extended, updated and improved under the name of NETFI-2. NETFI-2 can emulate the effects of SEU and SET in a CUT using only a single FPGA. Also, NETFI-2 permits an inject faults approach without modifying the source code of the CUT to model the faulty behavior. In the second framework presented in Chapter 5, we have proposed a fault injection methodology named NoCFI. NoCFI is a hybrid fault-injection framework that allows emulating multiple faults such as MBUs and

SEMTs taking adjacent cells into account. In other words, NoCFI makes use of the layout information of the circuit under test to emulate the effects of multiple failures in their neighboring cells. Furthermore, NoCFI is a simulation-based and emulation-based fault injection methodology which means that it can be used to inject faults in both HDL-simulators and FPGAs-based architectures.

1.2.1 NETFI-2

In Chapter 4, we introduce the NETFI-2 fault injection methodology. NETFI-2 allows for unique hardware resources a highly efficient fault injection campaign as well as a very accurate error rate estimation. This is achieved by a dedicated embedded campaign controller in charge of injecting SEUs and SETs in sequential and combinatorial blocks with configurable granularity. Because of its configurable granularity, NETFI-2 can determine which specific component is responsible for each error observed in the output of the CUT.

So, in Chapter 4 the following contributions were made:

- we have introduced a new version of our old fault injection method. This new version is more friendly to use and also presents more accuracy in terms of radiation-effects emulation.
- The whole fault injection campaign can be done using only one FPGA. It means that both the controller of the fault injection campaign as well as the CUT are prototyped inside the same FPGA.
- We have analyzed the effects of transient faults in the emerging stochastic computer technology such as Bayesian Machine (BM) and Support Vector Machine (SVM). Moreover, a real radiation beam campaign was performed in the SVM architecture. This campaign was conducted using a proton accelerator from GENEPI2 [118], and its results were compared with the ones provided by our fault injection tool.

To the best of our knowledge, this is the first work that presents a comprehensive analysis of the effects of radiation on the stochastic computer circuits. Some parts of this work were published in [26, 27, 110].

1.2.2 NoCFI

Although we have introduced a generic fault injection methodology in Chapter 4, that method does not take into consideration some issues relevant for the study of the effects of radiation on integrated circuits. So, in Chapter 5, we are more specific and propose a method to emulate soft-errors in NoCs named NoCFI (Network-on-Chip Fault Injection). This method supports

the emulation of single faults as well as multiple faults by taking cell adjacency into account. It is possible because NoCFI combines booth ASIC-based and FPGA-based flows to inject faults. The idea is simple. First, NoCFI performs place&route for the NoC under test in order to provide netlist and layout information. Then, NoCFI manipulates the gate netlist provided by the ASIC design flow and inserts additional circuits (i.e., saboteur) in the gate netlist. Finally, NoCFI uses the modified gate netlist as input for the FPGA design flow to emulate multiple faults in the Networks-on-Chip.

In summary:

- We have proposed a methodology that can be used to identify vulnerable cells nodes in the NoC's design and allow the classification of placement strategies of fault tolerant ASIC designs.
- We have developed a tool to manipulate the netlist provided by the place&router and translate it to a netlist that can be used by the FPGA design flow.
- We have presented a hybrid methodology that can be used as simulation-based and/or emulation-based taking into account the layout position of the NoC's cells.
- We have used a generic 5-stage pipeline 2D-NoC (Two-Dimensional Networks-on-Chip) in order to evaluate our methodology.

A preliminary version of the NoCFI was published in [29].

Part II

THREE-DIMENSIONAL NETWORKS-ON-CHIP

Chapter 2

A Soft-error Resilient Route Computation Unit

2.1 Introduction

One of the biggest concerns raised by the VLSI community regarding recent and future Networks-on-Chip (NoCs) designs is the continuous decrease in feature size and its impact on reliability. In fact, as the feature sizes of integrated circuits decrease aggressively, combinational logic becomes more susceptible to transient faults [85]. Specifically, *soft-errors* provoked by defects, radiation particles, or cross-talk noise, were generally masked and thus disregarded in older technology. However, as operating voltages become lower and clocking frequency increases, the design of integrated circuits with technology node below $0.25\mu m$ requires of particular attention to soft-error effects [108]. Indeed, NoC routers are not the exception since they have several combinational logic elements. In fact, the probability of the occurrence of a soft-error is higher in 3D-NoC routers which are more complex than 2D-NoC routers because of the increasing number of connecting ports. Therefore, the study of 3D-NoC routers reliability towards soft-errors becomes mandatory for future large-scale integration of dependable System-on-Chips (SoC).

Components inside a NoC router are typically structured into two interacting modules, the *control path* and the *data path* [59]. Soft-errors occurring in the data path can affect the data encoded in the packet. Fortunately, this type of fault is easy to detect and correct through existing error detecting and error correcting codes [125]. By contrast, faults in the control path are harder to detect and correct, and may leave the network in an inconsistent state, ultimately causing the entire chip to fail.

One of the most critical component in the NoC's control path is the Route Computation Unit (RCU), as it is the one responsible for selecting the next output port (i.e. direction) that a packet must take at every hop. Also, most recent proposal in the area of NoCs favor the use of adaptive routing for fault-tolerance, load balancing, etc., requiring more sophisticated routing

decisions [18,103]. This makes the Route Computation Unit significantly more complex, which can be expected to increase its vulnerability to soft errors. If the RCU fails, packets may be forwarded to wrong outbound ports (i.e., *misrouting*) eventually leading to deadlocks (cyclic dependencies between packets) or packet loss. It is therefore mandatory to provide some level of protection against routing errors.

In this chapter, we propose to enhance the reliability of 3D-NoCs by detecting and correcting errors provoked by transient faults in the RCU. The primary characteristic of our method is the ability to reliably and quickly *detect* misrouting based on a combination of fault tolerance techniques. Those techniques are called double-sampling and complementary illegal turn detection method. And they are combined into a specific hardware unit called *fault detection circuit*, which can make decisions in order to recover from faults. The second characteristic of our method is the ability to *correct* misrouting either by the reuse of the route computation samples provided by double-sampling or by directly rerouting the in-transit packet. In order to validate the robustness of the proposed solution, we have used a method to simulate transient faults using NoC's gate-level netlist. This method of fault injection is presented in more details in Chapter 5. Finally, a thorough evaluation on a partially vertically connected 3D-NoC is performed to demonstrate the increased resiliency, and to estimate the area and power overhead of the proposed fault-tolerant routing computation unit architecture.

2.2 State-of-the-art

Previous works have used *spatial redundancy* (i.e., execute parallel routing calculations) to deal with faults provoked by soft errors in NoCs. For example, the authors in [30] proposed the BulletProof router that employs N-modular redundancy (NMR) techniques to provide fault tolerance. The work in [127] proposed a fault-resilient routing unit for NoCs based on of a single and simplified redundant computation unit operating side-by-side with the RCU. While the RCU supports a fully adaptive routing algorithm, the redundant unit only supports limited paths, but it is only activated when errors are detected. In general, spatial redundancy approaches are expensive, as they require more silicon area than the baseline router.

Another approach to fault tolerance in NoCs is based on *temporal redundancy* (i.e., repeat routing calculations). Authors in [34] applied temporal redundancy in fully-connected 3D-NoCs. The authors in [60, 128] propose a mechanism to detect faults based on illegal turns in the chosen packet path. The fault detection is done at the neighboring routers which repeat a simplified route computation using input parameters provided by the previous router. If the selected direction is valid, the packet handling process continues, if not, either a new route is calculated or the packet is dropped. In general, temporal redundancy solutions incur in increased packet processing time.

Hybrid spatial and temporal approaches were also explored. For example, authors in [22]

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

Packet retransmission techniques were also proposed to trigger recalculations when necessary. In [89], the authors proposed a mechanism to detect and recover from transient faults through the analysis of the requested output port. When the RCU request an invalid output port, the router triggers a new routing computation to correct the error (i.e., rerouting). If the fault cannot be detected, the next router in the path will detect the fault and send a negative-acknowledgement (NACK) message to the previous router unit asking for recalculation and retransmission. In a similar approach, the FoReVer framework [88] presents a method to detect and recover lost, duplicated, and misrouted packets from routing errors. Since FoReVer is based on End-to-End detection and recovery, dealing with soft errors requires retransmission of the whole packet. In general, retransmission-based recovery mechanisms require additional retransmission buffers.

The work in [20] proposed to detect misrouting either in the faulty router or in the next-hop based on turns forbidden by the routing algorithm and blocked output ports (non-connected ports in the edge). The method is based on the fact that small components of the RCU are unlikely to evidence soft-errors allowing to simplify the correction. Thus, faults occurring at the address comparison stage are not detected. The primary correction mechanism is to reroute the packet. If faults are not detected in the faulty router, a dedicated network interface allows to re-inject the packet as a new packet in the NoC from the local router. Although re-injection minimizes packet drop, memory overhead is required to store the whole packet.

Our idea with this chapter is present a fault tolerant mechanism to detect and correct *all* routing errors before the packet leaves the router. And with this, preventing deadlock in the next hops. To achieve this goal, we make use of an additional fault detection circuit based on double-sampling and a rerouting technique to recover from soft-errors. Furthermore, and in contrast with previous 3D-NoC reliability works [34], we focus on the compatibility with partially and vertically connected 3D-NoCs [6, 19].

2.3 3D-NoC Architecture Background

In this section, a general overview of NoC designs is provided. NoCs are defined by many characteristics such as the network dimension, topology, switch architecture, switching technique, and flow control. These characteristics have a direct impact on performance, latency, and power consumption. Although there are a lot of different architectures and topologies concern-

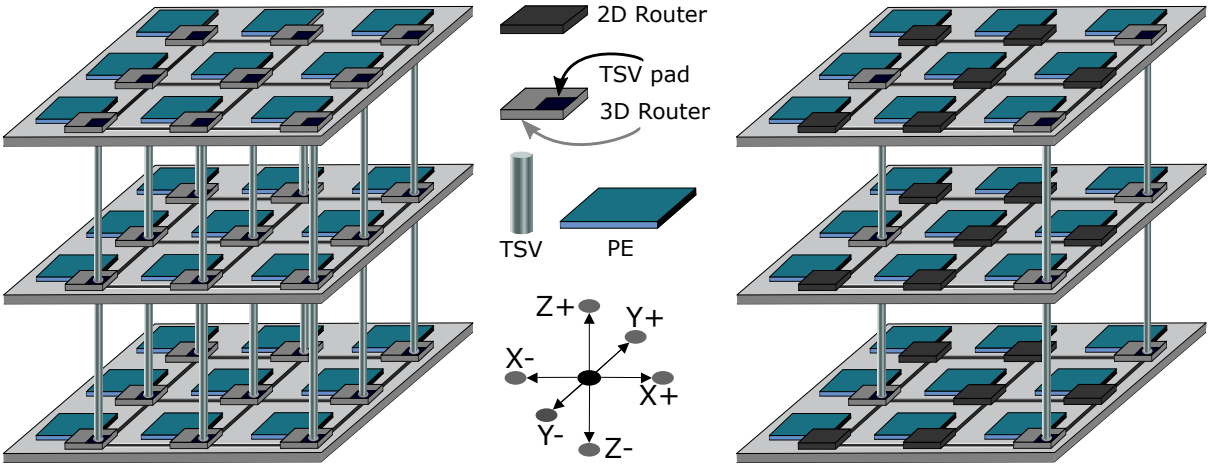

Fig. 2.1: Three-dimensional Networks-on-Chip (a) fully connected and (b) partially connected.

ing NoCs, we focus only on the NoC's characteristics that are most pertinent to this thesis. In particular, we have described a three-dimensional Networks-On-Chip architecture.

As said before, the transmission delay between distant routers is significantly increased when the size of a 2D-NoCs scales up. It results in lower performance and higher power consumption. So, to avoid such characteristics, the idea is moving toward the concept of 3D integrated circuits, where layers are vertically stacked, instead of growing in two dimensions. In this case, a 3D-NoC can be viewed as many 2D-NoCs layers that are vertically stacked, such as a sandwich, and connected through vertical connections. It means that in addition to planar connection there are also the vertical connections which permit Up and Down communications intra-circuit. Figure 2.4 (a) shows an example of a fully connected three-dimensional networks-on-chip where it is possible to see that all the routers are vertically connected (in this thesis, we have adopted TSVs such as the technology to vertical connections). It means that in a fully connected 3D-NoC all routers can send and receive messages in the planar, Up and Down directions. This type of 3D-NoC topology presents high performance since the packets can use the minimal path to achieve its destination. Also, in case of failures in one router, it is possible to select any other router to send packets vertically to its layer of destiny. However, fully connected 3D-NoCs presents some disadvantage such as it needs more virtual channels because they have more connection than 2D-NoCs. Also, the cost of TSV's manufacture is expensive and the reliability of the circuit decrease as the number of TSV increases [45].

Due to the high manufacture cost of vertical connections, the partially and vertically connected solution has gained attention as an alternative to the use of the 3D IC technology [39]. Figure 2.1 (b) shows a 3D-NoCs which is built using only a few vertical connections in its architecture. This type of architecture is called partially connected three-dimensional Networks-On-Chip. Although the use of partially connected 3D-NoCs decreases the manufacturing cost of the 3D ICs, the reduced number of vertical connections brings new challenges for the 3D-NoCs architecture. In particular, the routing algorithm and the flow control of the NoCs tend

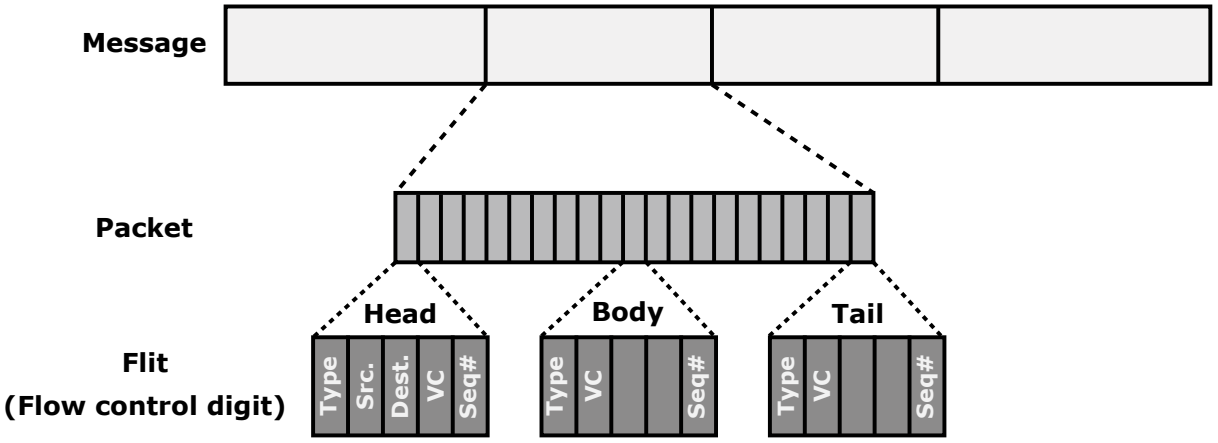

Fig. 2.2: Packets and Flits definition

to grow in complexity since the number of valid paths is directly affected by the number of vertical connection. Moreover, the task to achieve a path for a packet can be yet more complicated when permanents and/or transient faults happen inside a router, which can, in some cases, invalidate all the NoC operation. With this issue in mind, in this thesis are proposed fault-tolerant techniques to improve the reliability of the partially connected 3D-NoCs architecture. But, before detailing the architecture of partially connected 3D-NoC used in this thesis, in next subsection 2.3.1 we firstly introduce some definition, properties, and control flow about the NoC operation. So a brief background about the partially connected 3D-NoCs architecture is presented in the section 2.3.2.

2.3.1 NoC switching Properties

The switching technique determines how the NoC's resources (routers, buffers, links, etc.) are allocated to a message route determined by the routing protocol between the source and the destination nodes. In this thesis, a packet switching technique is adopted due to the fact that it has better scalability for NoCs and is commonly used in most typical NoCs implementations. In an NoC based on packet switching, the link between routers is iteratively allocated until the destination node is reached. For efficient utilization of router resources, a message to be sent over the network is broken into discrete sized packets as depicted in Figure 2.2. The packets are further broken down into smaller units of fixed bits called flits (flow control information units). The flits are the smallest unit of data sent between different nodes in the network and are of three types:

1. Head flit: Each packet contains a single head flit which carries information about the destination of the packet.

2. Body flit: Each packet can have several body flits which carry the actual data to be sent between the different network nodes. The body flits are contained between the head and

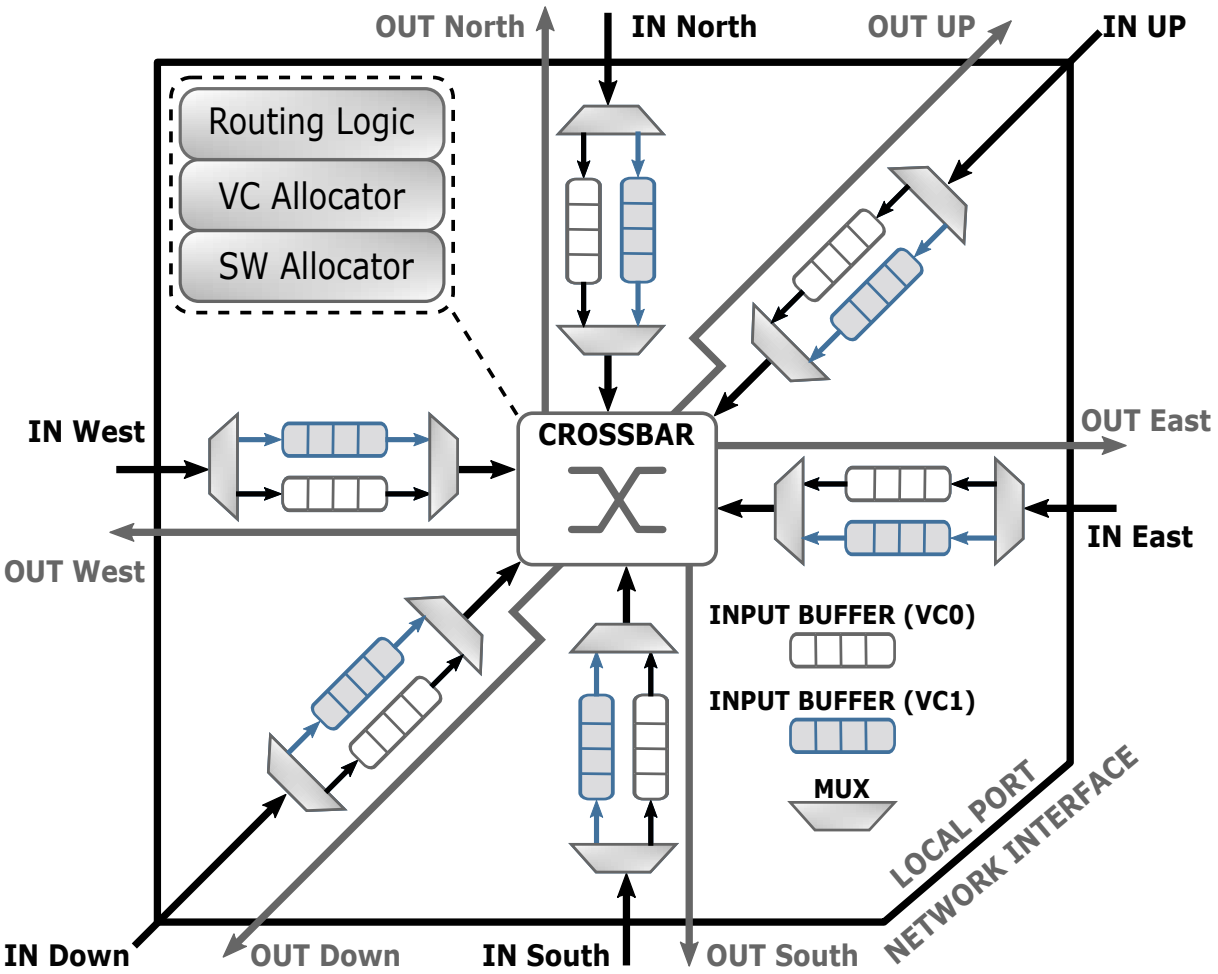

Fig. 2.3: 3D-router architecture with double virtual channels for each direction.

tail flits.

3. Tail flit: Each packet contains a single tail flit which signifies the end of the packet.

In addition to the information enumerated above, each of the flits also carry information of the type of flit, the virtual channel (VC) which the packet is being routed on, and its sequence in the packet. Furthermore, the head flit is used to reserve an output port of the router and the reservation is maintained till the tail flit is transmitted. This technique has the advantages of increased throughput, lower latency and smaller buffer sizes.

Although the packet switching technique presents a good solution for hardware-implementation in the VLSI design, some issues must be addressed in the NoCs project. In particular, three issues/properties must be **avoided** for any usable NoC architecture:

- **Deadlock** is a situation in which the packets are involved in a circular dependency which cannot be resolved. Deadlock results in the packets making no progress towards the destination node. The deadlock freedom can be provided by the characteristic of the routing protocol or by using costly deadlock detection and resolution logic.

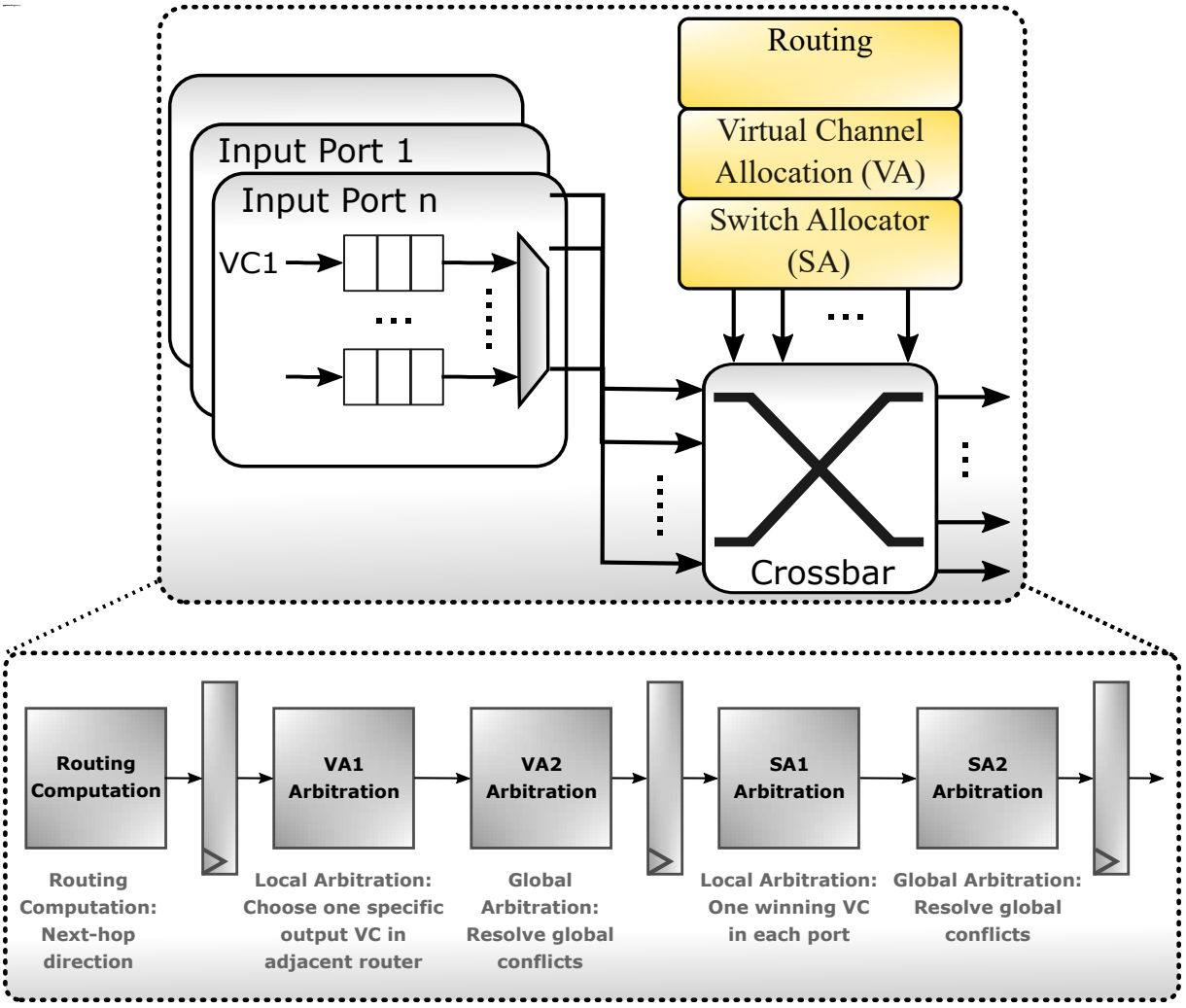

Fig. 2.4: Generic Router architecture with pipeline

- **Livelock** is a situation in which the packets wander about in the network without ever reaching the destination. Minimal and deterministic routing protocols are livelock free, while non-minimal routing protocols can route packets into paths that do not lead them towards the destination.
- **Starvation** is a situation in which a packet is starved of resources and never gets serviced in a router. Starvation is an important aspect in the design of router arbitration mechanisms.

2.3.2 Partially connected 3D-NoC architecture

In this thesis, we consider a partially vertically connected 3D-NoC mesh architecture, wherein the routers include, in addition to the usual five ports (East, West, South, North, Local), either an Up port, a Down port, or both (i.e., 5, 6 and 7 port configuration), as shown in Figure 2.3. It means that each layer consists of a mixture of classic 2D routers including only 5 ports and 3D

routers having either 5, 6, or 7 ports as we can see in Figure 2.1. The Up and Down ports of the router are connected vertically using Through-Silicon Via (TSV). Also, each router is identified by its coordinates (X, Y, Z), where X identifies the column, Y the row, and Z the layer.

A typical input-buffered wormhole router such as the one shown in Figure 2.4 is adopted. In this architecture, the router consist of a seven input port, seven output port, two virtual channels per port, and a control logic distributed in four-stage pipeline. The control logic of the router comprises of Routing Computation (RC) unit, Virtual Channel Allocation (VA) unit and the Switch Allocation (SA) unit. A central crossbar (XB) connects the input and output ports of the router. So, routing a packet is performed in four stages:

1. **RC Stage**: This is the first stage in the pipeline and is active upon the arrival of a head flit into the router. Based on the destination information available in the head flit and the used routing protocol, the RC unit determines the output port of the current router through which the head flit will leave. This stage remains idle for body and tail flits.

2. **VA Stage**: This is the second stage in the pipeline and is active upon completion of RC stage. This stage also operates only on head flits. Figure 2.4 shows the architectural block diagram of a two-stage separable virtual channel allocator (i.e., *VA1* and *VA2*). In the first stage, based on the result of RC, every input VC with a head flit arbitrates for an empty VC at the downstream router. In the second stage, head flits across different input VCs that have been allocated the same VC at the downstream router compete with each other. The input VC that wins the arbitration in the second stage is allocated to the VC at the downstream router.

3. **SA Stage:** This is the third stage in the pipeline and is active for head, body and tail flits. SA unit is responsible for determining which input VC from an input port gets to transmit a flit through the crossbar in the next cycle. Figure 2.4 shows the architectural block diagram of a two-stage separable switch allocator (i.e., *SA1* and *SA2*). In the first stage, the SA unit decides which VC of an input port gets to transmit its flit through crossbar. In the second stage, competition between different input VCs trying to gain access to the same output port of the crossbar is resolved. The input VC that wins the arbitration in the second stage gets to transmit its flit through the crossbar in the next cycle. Unlike routing computation and virtual channel allocation, switch allocation stage is active for head, body and tail flits

4. **Crossbar Stage:** This is the final stage in the pipeline and is active for head, body and tail flits. Crossbar connects the input and output ports thus facilitating flit traversal from a VC of an input port to an output port. Figure 2.4 shows the architecture of a crossbar. SA unit is responsible for generating control signals to the multiplexers in crossbar. Input output port connections of the crossbar are configured every cycle based on the winners in SA stage.

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

2.3.3 Routing Computation Unit

In order to prevent deadlocks and reduce congestion, existing routing computation solutions for irregular 3D-NoCs employ virtual channels [39], [19]. As said before, virtual channels are buffers inside the input port of the router that allow storage flits from different packages. The idea is to multiplex a physical channel using multiple virtual channels (VCs) in the input port. In this sense, we are interested in the use of VCs grouped by the concept of "Virtual Network", which was proposed by [32]. A Virtual Network (VN) is a logical network projected on the physical network that uses fixed virtual channels which can be selected by the routing algorithm. To better clarify the concept of VN, lets us considered the example illustrated in Figure 2.3. In this example, each router has two buffers (two virtual channels) on the physical channels, and also it has two VNs (one for Up direction and another for Down direction). Each VN is assigned one virtual channel, and each VN has its own routing algorithm. It means that messages whose destination is in the UP direction of the source are injected into the VN0, and messages whose destination is the down direction of the source are injected into the VN1. In other words, we defined by two sets of VN that the packets which need to go to North and East can only use the virtual channel 0, and the packages that need to go to South and West can only use the virtual channels VC1. The use of several VNs thus increases the diversity of the actual routes.

So, in addition to the output port, the RCU also selects a virtual network (VN) number. This number is used by the VA to determine the set of VCs that can be allocated to the packet as described in [19]. Each VN can have its own routing algorithm and rules that avoid creating cyclic dependencies between packets. In this context, the route computation, is not only responsible of selecting the next output-port but also the VN.

The route computation is performed in three stages [20]. The first stage is used to compare the relative position of the packet and its destination, then the second stage adds turns constraints which are based on the routing algorithm. Finally, the third stage select only an output port and a virtual network to be attributed to the packet. Those stages are described in more details below:

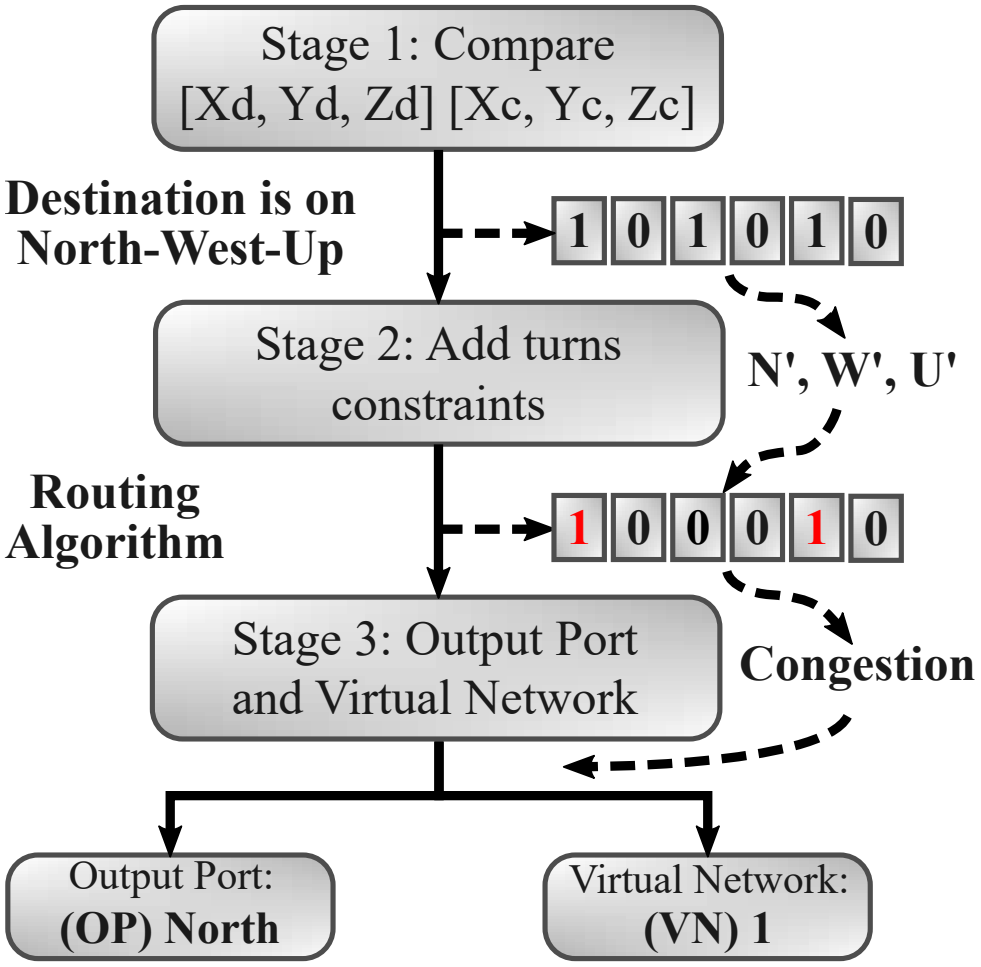

Fig. 2.5: An example of the three stages of the routing computation unit

1. In the first stage, the relative position of the packet's destination (Xd, Yd, Zd) is compared with the address of the current router (Xc, Yc, Zc) to produce a candidate direction vector [N', S', W', E', U', D']. In the direction vector, one or more signals may be active. For example, if the destination is in the North-West-Up quadrant, then N', W', and U' signals will be enabled and the vector will be configured as [1, 0, 1, 0, 1, 0] as shown in result of Stage 1 of Figure 2.5.

2. The second stage is used to add turn constraints according to the implemented adaptive routing algorithm. In this case, a vector [N", S", W", E", U", D", L"] representing the possible or legal directions is computed for each routing algorithm presented in each VN. In this case, since the algorithm we adopted in this example takes the positive direction (North, East, and Up) first, the stage 2 will disable the West direction from the bits vector. So the results after the stage 2 will be a vector such as [1, 0, 0, 0, 1, 0] as we can see in the Figure 2.5.

3. Finally, in the third stage, an output port and a VN are selected among the legal routes by taking into account information such as congestion. So, continuing our example illus-

trated in the Figure 2.5, the routing algorithm will select the North direction because based on the congestion, and also the routing algorithm will select the VN0 (Virtual Network number 0) because the destination is in the Up direction of the source.

2.4 Resilient Route Computation Unit

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

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

2.4.1 Detection: Double Sampling and Custom VC Allocator

The Double-Sampling (DS) is a method that permits to observe, at two different instants, the outputs of the combinational logic of each pipeline stage [82]. The main idea is to add a redundant sampling element (latch or flip-flop) to each output of the pipeline stages that need to be checked and clocking this redundant sampling element by a delayed clock signal.

Figure 2.6 illustrates the basic operation of the double-sampling mechanism used in this work. We propose to use the rising edge of the clock as latching event of the regular flip-flop and the falling edge of the clock as latching event of the redundant sampling element to get the **output port (OP)** and **virtual network (VN)** computed by the RCU ([OP,VN]). Since the RCU is performed in one clock cycle, and the double-sampling is running in parallel with RCU, this technique does not result in any path delay.

The Sample_1 and Sample_2 are obtained in different instants, as shown in Figure 2.6. Those two samples are compared by the additional Fault Detection Circuit (FDC) in the next cycle, after route computation. If the result of comparison between these two samples is different, then a fault has affected one of the samples. Therefore, both samples need to be analyzed with the purpose of determining which one is faulty. In our implementation, Sample_1 is initially verified, followed by Sample_2. On the other hand, if the result of comparison between these two samples is equal, maybe because they are fault-free or the same fault affecting both samples, only Sample_1 is selected to check if there is error or not.

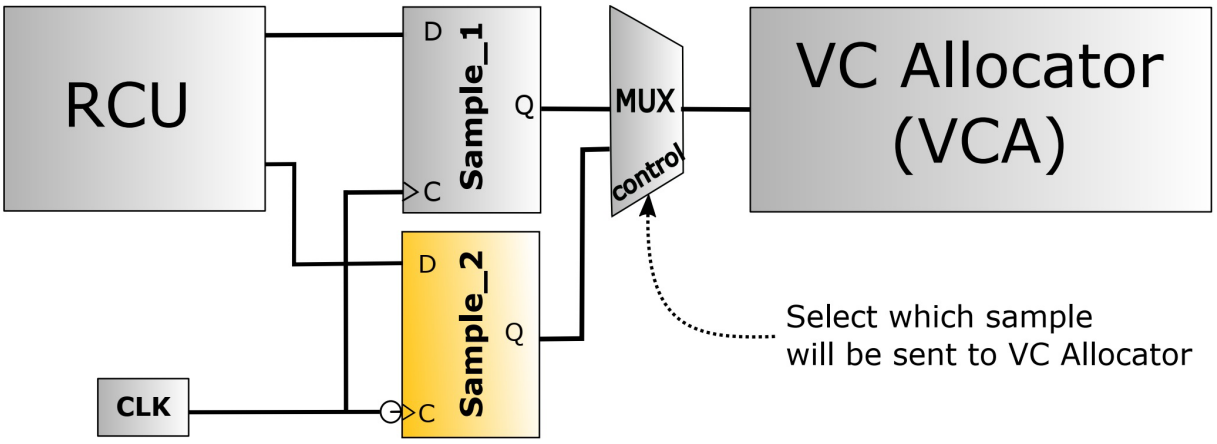

Fig. 2.6: A Double-Sampling implementation.

2.4.2 Detection: Custom VC Allocator

The detection of an incorrect route vector ([OP,VN]) can be performed directly by the VC Allocator (VCA) or by an extra Fault Detection Circuit (FDC), which runs in parallel with the VCA, as shown in Figure 2.7.

Because the VCA includes a table indicating the possible input-output connections according to the routing algorithm, it can detect illegal output port and VN requests with minor modifications. For example, if a request demanding for a blocked output port (i.e., the requested output port is unconnected because it is located at an edge of the 3D-NoC), the VCA can declare a routing error. Another fault that the VCA can directly detect is when the requested turn is forbidden by the VN parameter of the local router. These two types of faults can be easily detected by simple modifications in the VCA circuit in the NoC router.

For faults that do not involve illegal turns, more sophisticated approaches are needed. This is the case when the output port and VN are both valid, but the packet needs to make illegal turns at later hops to reach its destination. To deal with these faults that cannot be detected by the VCA, an extra Fault Detection Circuit (FDC) is included in the NoC router, as shown in Figure 2.7. To this end, the FDC checks three conditions. First, it compares the Zd and Zc to know if the packets which need to go in the Up direction were wrongly selected to go Down, or vice versa. Additionally, it checks if the packets are directed to a local port, in this case, the FDC compares the position of the current router with the destination of the packets. Second, the FDC checks if the direction adopted by the packet leaves to an illegal turn in the upcoming packet path. For example, if a routing unit using negative-first algorithm selects a valid direction, e.g. North which is a positive direction, but the destination is located West to the current router, this turn is declared illegal, as the packet will be forced to take the West (negative) direction at later hops after the North, leading to deadlocks. Third, it checks the two samples provided by the double-sampling and selects which of them must be finally used by the VCA.

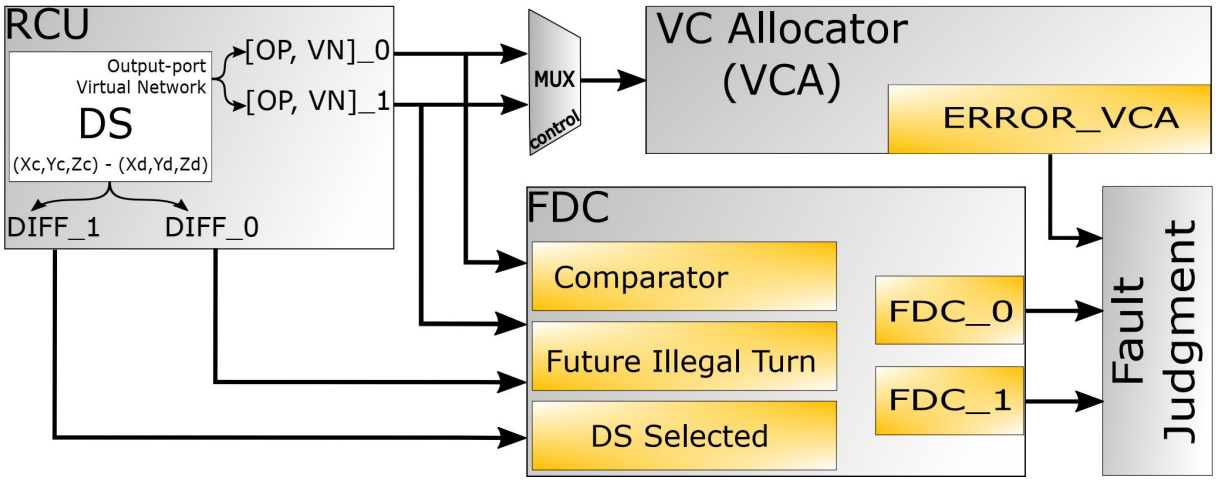

Fig. 2.7: Fault detection Circuit - FDC.

2.4.3 Detection: Fault Detection Circuit

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

2.4.4 Correction: Rerouting

Unfortunately, repeating the route computation process is not simple since one RCU is shared among all VCs in the same input port. This means that each port is able to do one route computation per clock cycle. Evidently, a first alternative for this limitation is to add an extra RCU per VCs per input port. However, this approach increases the area and power overhead of the NoC. To avoid replicating the RCU for each VC, we propose a simple rerouting mechanism that uses the existing RCU. As illustrated in the Figure 2.8, the idea is to ensure that a limited number of new packets enter an input port in which one or more packets have requested rerouting. The solution consists in preventing packets from entirely leaving the input port, ensuring the buffers do not get available to receive new packets until all packets have been rerouted [20]. If a new

packet header is received at the input port, it is routed by the RCU, otherwise one of the packets requesting rerouting is allowed to use the RCU to be rerouted. Because a finite number of new packets will be received, our method effectively guarantees that all requesting packets are eventually rerouted.

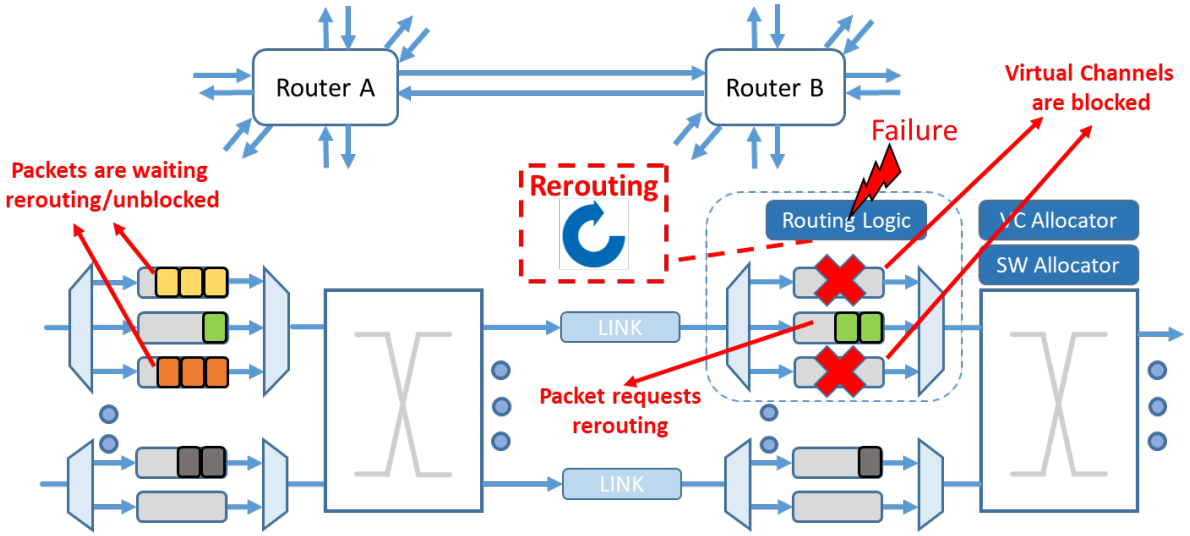

Fig. 2.8: A Rerouting Scheme implementation.

Two rules are created in order to block the input port with minimal impact in the overall network flow. If there is one packet in the input port asking for rerouting, a flag is set to block this port in order to stop receiving new packet headers. Indeed, this rule allows other parts of the packet which are not the header to still be accepted in the local port. In this case, if we have no header coming in the input port, rerouting is performed in one of the VC that requires rerouting. If more than one VC is requesting rerouting at the same time an arbitration is done to choose which one will finally perform the calculation first.

2.5 Fault-Injection Experimental Procedure

In order to analyze the reliability of the resilient RCU proposed in this work, we introduce a fault injection methodology that can mimic, with high accuracy, the effects of transient faults in the NoC's gate-level netlist. The most relevant characteristic of the resulting tool-chain is its ability to inject faults without modifying the original Hardware Description Language (HDL) code of the NoC. Instead of editing the original HDL design, the original library provided by the design kit used to make the synthesis of the design was modified. In particular, we have modified the design kit FD-SOI $28nm$ technology from ST-Microelectronics to inject faults in the NoC's gate-level netlist. Figure 2.9 illustrates the proposed work-flow.

In summary the fault injection campaing was perfromed in four steps:

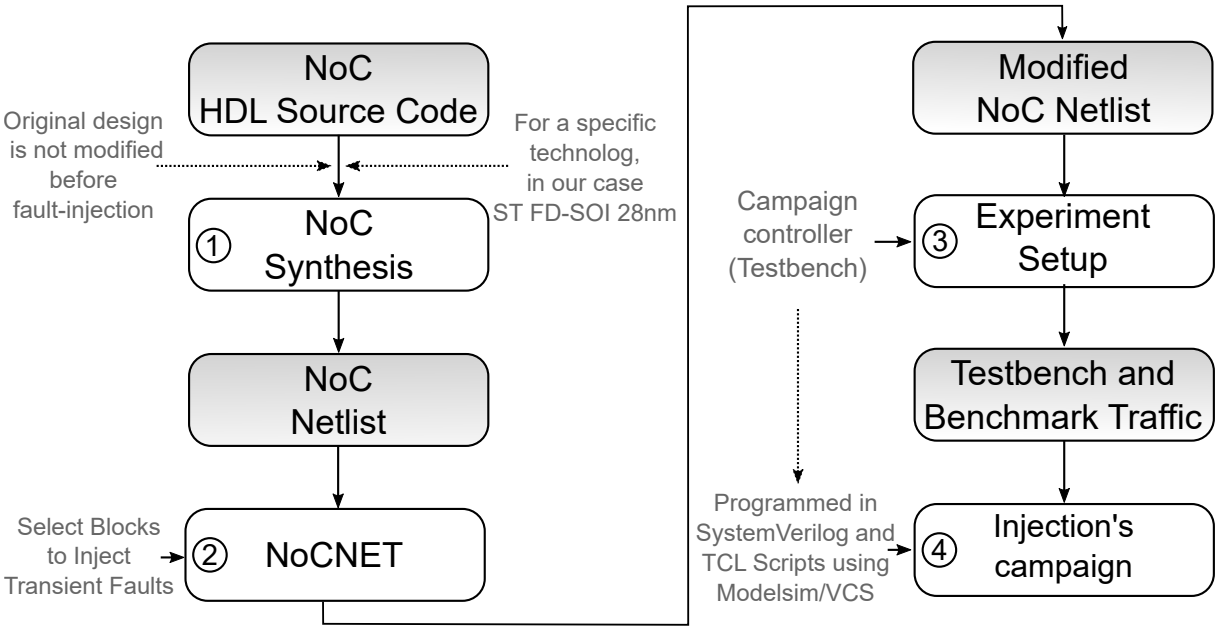

Fig. 2.9: Fault injection methodology

Initially, the HDL description of the NoC is used to obtain a first synthesis in Step ① . This synthesis was performed using Synopsys Design Compiler, as it allows to export a gate netlist based on the technology adopted, in our case ST FD-SOI $28nm$. In Step ② , the netlist is used as input for the NoC NETlist (NoCNET) tool. The output of NoCNET is a modified (but functionally equivalent) netlist with a large number of extra input signals used to access all logic blocks of the NoC to inject faults. The resulting synthesis of the modified netlist includes some additional combinational circuitry to the design. In the case of transient faults, NoCNET modifies all the logic gates of the netlist by simply adding an extra multiplexer at the output to select the appropriate value (erroneous or correct). In Step ③ , a campaign controller is integrated within the modified netlist. The campaign controller is a HDL testbench implemented in SystemVerilog that is in charge of managing the fault injection campaign by being directly wired to the NoC modified by NoCNET. To this end, the netlist obtained in Step ② is attached to the testbench by using the modified library (ST FD-SOI $28nm$). Finally, the experiment in Step ④ can be directly executed using simulator tools like Modelsim (Mentor Graphics) or VCS (Synopsys). The whole fault-injection campaign (including the post-processing of the results), can be conveniently encoded in the testbench and automated by means of Tool Command Language (TCL) scripts. By accessing the interfaces connecting the NoC, fault injection signal and network interfaces, the testbench can efficiently execute several iterations of fault-injection experiments randomly selecting the fault points and running different benchmarks traffic.

2.6 Evaluation And Analysis

In order to analyze the latency, hardware cost, and reliability of the proposed resilient RCU, we have extended the cycle-accurate Netmaker library [77] to support 3D-NoC router architectures. Netmaker is a library of parameterizable and synthesizable NoC routers written in SystemVerilog. This library was used to implement the detection and correction mechanism described in this Chapter, as well as the 3D-NoC router baseline described in [19].

The executed fault injection campaigns were performed in a partially and vertically connected 3D-NoC composed by 64 routers distributed in a 4x4x4 mesh architecture. The 3D-NoC planes are connected by 4 TSV pillars (25% vertical connections). This 3D-NoC routers are based on 4-flit deep FIFOs, and a packet size fixed in 5 flits. The routing algorithm uses the same configuration of VCs presented in [19]. When several candidates are available, the routing algorithm selects the least congested output port, based on a local congestion metric. For each simulation, the 5-flit synthetic packets are injected and wait until 100000 of them are received under a pessimistic fault injection rate of 5%.

We have tested this 3D-NoC built with three different NoC routers. The first circuit is the baseline router here termed ROUT3D. The second circuit identified as ROUT3D-TMR (Triple Modular Redundancy), is a version of the ROUT3D with TMR-hardened RCU in each port of the NoC routers. Finally, the third circuit implements the architecture described in Section 2.4.1. We refer to this circuit as ROUT3D-FDR (Fault detection circuit, Double sampling, and Rerouting).

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

2.6.1 Latency Results

The performance of the proposed RCU architecture is estimated through the average packet latency under three different traffic patterns (Uniform, Bit-complement, Shuffle). The resulting performance comparison is shown in Figure 2.10 when there are faults in the ROUT3D-FDR and no faults in the ROUT3D baseline. The degradation in performance is noticed because to recover from misrouting the VC, the same input port needs to be blocked to avoid the reception of new headers while executing the rerouting routine. The tendency is that packet's delay increases with the injection rate. This can be explained by the fact that a higher injection rate means a higher probability that new packets enter the router and use the routing unit instead of the rerouting mechanism.

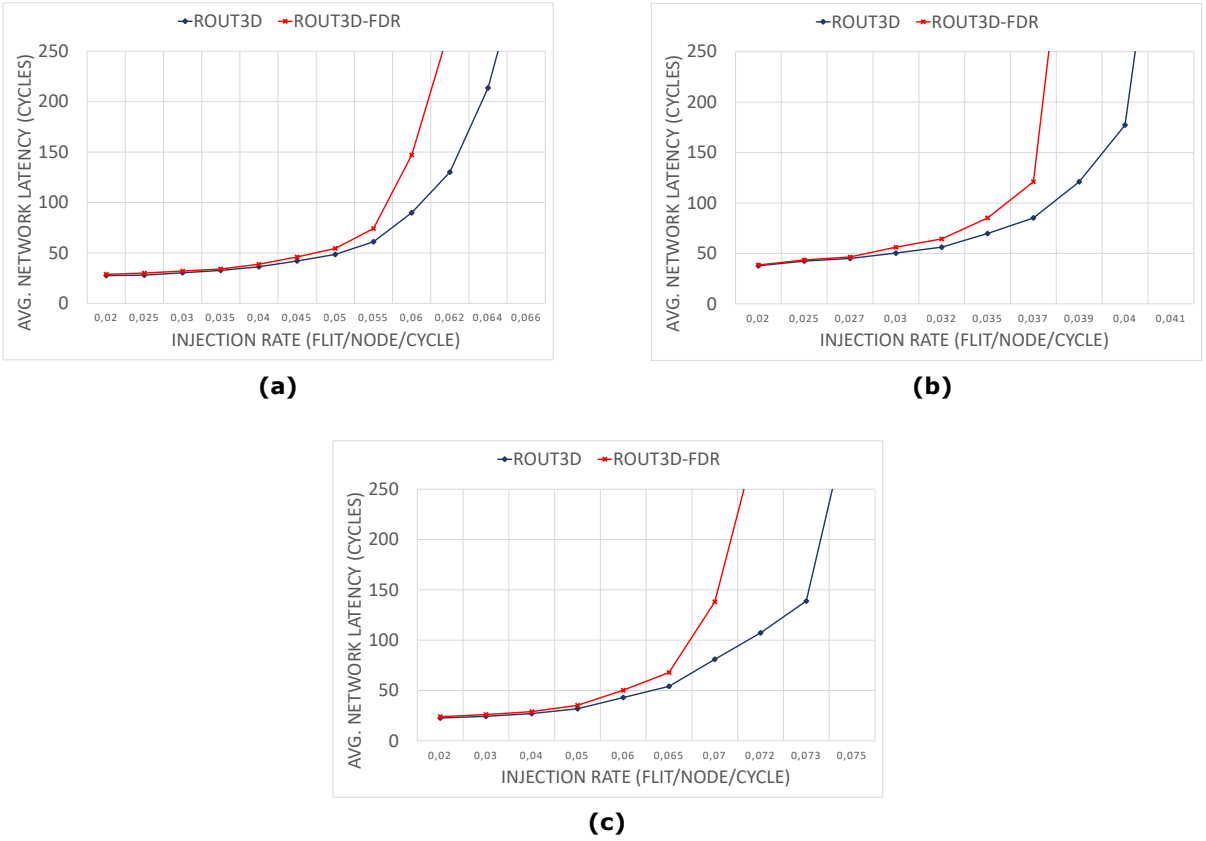

Fig. 2.10: Latency of ROUT3D-FDR and baseline ROUT3D under (a) uniform traffic, (b) bit complement traffic and (c) shuffle traffic.

2.6.2 Hardware Synthesis Results

In order to evaluate the hardware overhead of the proposed resilient Routing Computation Unit (RCU) solution, we have performed two syntheses. In the first synthesis, we estimated the area and power overhead when all designs are setup to work with an operating frequency of 1GHz, a power supply of 1V, and a commercial ST FD-SOI $28nm$ library. In the second synthesis, the tools are configured to achieve the maximum operating frequency. It is possible because the tools explore some configuration for a maximum operating frequency taking into account the critical path delay. The results for both syntheses are summarized in Tables 2.1 and 2.2. The three types of routers previously described were considered: 5-port 2D routers, 6-port 3D routers with one vertical connection, and 7-port 3D routers with both Up and Down vertical connections.

The area and power overhead showed in the Table 2.1 put in evidence the low hardware overhead of the proposed method. We can see that for the 5, 6 and 7-port routers from ROUT3D-FDR, the area overhead increases around 4% while the ROUT3D-TMR increase by 12%. It is important to note that while the ROUT3D-FDR needs only an additional sample (double-sampling) to register the results of RCU and an additional circuit FDC for fault detection, the ROUT3D-TMR needs to triplicate all logic from RCU per input port as well as an additional

voter.

Table 2.1: Area and Power results for ROUT3D, ROUT3D-TMR, and ROUT3D-FDR

Size	**Area** (μm^2)			**Power** (mW)		
(# Ports)	ROUT3D	ROUT3D	ROUT3D-FDR	ROUT3D	ROUT3D	ROUT3D-FDR
5-Port	14945	16802	15564	5.8192	6.5628	6.2175
6-Port	19036	21309	19738	7.8411	8.5576	8.1612
7-Port	23888	26613	24681	9.6362	10.489	9.9963

In order to determine the maximum operating frequency that the 3D-NoC routers can achieve, we perform a sequence of syntheses increasing the operating frequency without time violation. The maximum operating frequency is shown om Table 2.2. As can be seen in this Table 2.2, the ROUT3D-TMR maintains practically the same maximum operating frequency while the ROUT3D-FDR decreases around 0.4% when both are compared to the baseline router. This is because the ROUT3D-FDR needs to decide which sample will be selected and if the packet needs to be rerouted after the FDC's analysis.

Table 2.2: Maximum Operating Frequency for ROUT3D, ROUT3D-TMR, and ROUT3D-FDR

Size	**Max. Freq** (MHz)		
(# Ports)	ROUT3D	ROUT3D	ROUT3D-FDR
5-Port	2380	2380	2372
6-Port	2300	2296	2290
7-Port	2200	2200	2190

2.7 Conclusion

In this Chapter, we have proposed a novel method to safely correct routing errors and deliver all packets to their destination without resorting to retransmission. It is possible, because we have included a mechanism to detect, mask, and then correct transient faults in the route computation unit. Also, using a single rerouting mechanism, it possible to detect all fatal routing errors before the affected packets leave the faulty router. It means that the proposed approaches guarantee that no packets are dropped because of misrouting. Misrouting is detected combining double-sampling technique with a virtual channel allocator (VC) optimization and an extra Fault Detection Circuit (FDC). The rerouting mechanism that we have proposed is simple and safe as it ensures that the routing logic is correctly shared between several requesters and that no starvation can occur. In other words, to minimize the implementation cost, the same routing logic is used to service all the rerouting requests of the same input port. Results obtained from

a partially and vertically connected 3D-NoC showed that all faults can be corrected locally at the expense of a minimal latency increment. Indeed, rerouting is triggered by an efficient detection approach. Moreover, the measured area and power overhead including double-sampling, rerouting, and fault detection circuit was 4.1% and 6.8% respectively. These results suggest that the proposed method is an appealing alternative to traditional TMR-based approaches.

Chapter 3

FL-RuNS: A High Performance and Runtime Fault-Tolerant Routing Scheme

3.1 Introduction

Among the vertical interconnection technologies, Through-Silicon-Via (TSV) has been accepted as one of the most viable technologies since it enables faster and more power efficient inter-layer communication across multiple stacked layers [12]. However, the TSV fabrication process suffers from lower yield [69,91]. The low yield of the TSV fabrication process is related to the chemical and mechanical properties of the material utilized. Specifically, 3D-ICs suffer from the conversion of thermal stress into mechanical stress during fabrication due to the difference in thermal expansion coefficients of the implementation materials [126]. Furthermore, the temperature variation between two layers can negatively affect the Time-Dependent Dielectric Breakdown and Thermal Cycling [45]. Also, the Electromigration [52, 104] can increase the resistance of the conductor which in turn increases the communication delay. In summary, employing a large number of TSVs degrades reliability, increases the manufacturing cost and causes area overhead.

To overcome some of these challenges, a minimum subset of the routers is connected to the upper/lower layers using TSVs while the routers in the same layer are connected using global links. This approach results in irregular 3D-NoC topologies, commonly referred to as Partially-Vertically-Connected NoCs. In fact, we are interested in a partially connected 3D-NoC mesh architecture such as the one shown in Figure 3.1. This type of architecture presents a limited number of vertical connections (TSVs), commonly known as **Elevators**. Also, it is composed of a combination of 2D- and 3D-routers distributed through the layers. Each router is responsible for connecting process elements as well as for sending and receiving packets through the network. Addressing a new routing algorithm for this type of architecture can be complex since the reduced number of vertical connections pose new challenges in terms of deadlock-free routing. A routing mistake can lead to a situation where packets are blocked

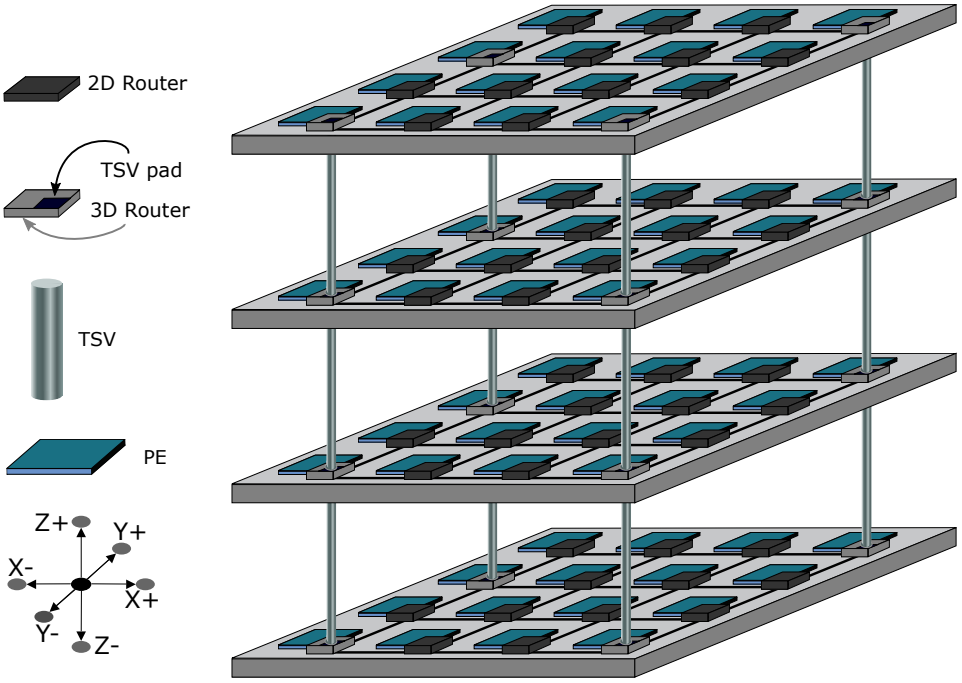

Fig. 3.1: 3D-NoC architecture vertically and partially connected

waiting for free resources in the networks (i.e., a deadlock situation). So, a definitive solution that presents high reliability, high performance, deadlock-free routing algorithm, and efficient TSV assignment for a partially connected 3D-NoC is still missing in the literature.

Based on these facts, we have previously proposed a routing algorithm, named First-Last [15, 17], which is able to tolerate TSVs failures only in the manufacturing phase. However, TSVs in 3D-ICs have become more fault sensitive, not only in the manufacturing phase; but, also during the operation time. Failures on TSVs can change the network configuration, increase the network latency, degrade the system performance, and ultimately reduce the overall 3D-NoC lifetime. In this context, many works have addressed the TSV failure problem adopting fault-tolerant techniques in the routing algorithm for 3D-NoC. However, as it is explained in Section 3.2, most of the already existing routing algorithms present one or more of the following problems: A large number of TSVs and Virtual Channels to recuperate from faults, specific routing rules that pose restrictions on the location and the selection of TSVs, and/or an offline mechanism to reconfigure the entire 3D-NoC after faults. All those techniques either significantly increase the hardware resources or need to drop packets in the reconfiguration phase until the network goes back to its stable condition again.

In this thesis, we propose a novel light-weight and efficient fault-tolerant routing scheme called **FL-RuNS** (**F**irst-**L**ast **Ru**ntime and Resilient 3D **N**etworks-on-Chip **S**cheme). FL-RuNS takes advantage of the high-performance of our routing algorithm First-Last [15], and combines it with a fault-tolerant routing scheme to overpass failures in the vertical link. FL-RuNS presents a reconfigurable 12-bit vector location for each router named *Elevator Bits*. The bits contained

in the Elevator Bits are used as a compass by the routing algorithm to search for a healthy elevator. Also, FL-RuNS includes a mechanism to propagate the TSV status and to update the Elevator Bits. Furthermore, FL-RuNS adopt an asymmetric virtual channel along the West, North, Up, and Down directions which are used as escape path to delivery packets in presence of runtime faults.

3.2 State-of-the-art

Fault-tolerant routing algorithms for 3D mesh architecture have been presented in HamFa [43], 4NP-First [90], AFRA [3], and HLAFT [2]. All these algorithms can tolerate a certain number of faulty vertical links in a fully connected 3D-NoC. However, they assume that all TSVs must be connected, which can greatly increase the manufacturing cost.

While some works have addressed fault tolerance in a fully connected 3D-NoC, only a few proposals have been made in the context of partially vertically connected 3D-NoC. The work in [112] proposes a fault tolerant routing algorithm able to tolerate faults occurring in the horizontal and vertical links. This solution uses three virtual channels for each port, and each router needs to save a table with all TSV's status. Although this technique presents a runtime fault-tolerant solution, the high hardware implementation cost limits its scalability.

The LBDR3D [81] is a framework that supports a variety of partially adaptive routing algorithms, based on the turn model [54], and can be reconfigurable to tolerate faults in horizontal and vertical links. LBDR3D uses a limited number of vertical bits to point to the nearest elevator. The nearest elevator is selected off-line based on the Manhattan Distance. It was proven deadlock- and livelock- free using the same method as the one used in Elevator-First and requires the same minimum number of virtual channels to separate between Upward and Downward flows. However, the method adopted in LBDR3D for assigning elevators to nodes is not inherently deadlock-free. Deadlock happens because LBDR3D uses a fully randomized manner to select an elevator when there are several elevators with equal Manhattan distance from a given node. The inconsistencies between several nodes can lead to the violation of the routing rules, potentially leading to deadlocks.

ETW [100] is an adaptive routing algorithm that requires one additional virtual channel along the Y dimension. In ETW, the packets have to take the East direction before moving toward West. This condition increases congestion in X dimension and results in lower performance. Most recently, the same authors proposed an adaptive routing algorithm called LEAD [101] that requires one more virtual channel than ETW along X direction. Unlike ETW, LEAD does not impose restriction rules on elevator selection and does not have the obligation in moving towards East. However, like ETW, LEAD does not include any strategy in case of runtime failures in the vertical link connection.

Elevator-First [39] is a deterministic routing algorithm for partially connected 3D-NoCs

which requires two virtual channels along X and Y dimensions. Elevator-First routes packets toward an elevator (vertical links) through the insertion of a temporary header containing the elevator's address. Although Elevator-First does not impose limitations in choosing elevator to transfer the packets to the destination layer, it cannot be reconfigured by any TSV failure which has occurred after the manufacturing process.

CoBRA [102] proposes some solutions to tolerate runtime failures using the propagation of TSVs status and a reconfiguration mechanism. Unfortunately, CoBRA poses some very limiting constraints on both the location and the selection of the elevator. It requires the existence of at least one elevator in both the east-most and the west-most columns. Using two virtual channels along the Y dimension to avoid deadlock, CoBRA initially forwards the packets to the East until finding a healthy TSV, and if no TSV is found at the east-most column, the routing algorithm is reconfigured to deliver the packets toward the West direction. The problem with this technique is that some packets are dropped in the reconfiguration phase until the network returns again to its stable condition.

The First-Last routing algorithm proposed in [15] uses a limited number of vertical bits and employs off-line reconfiguration to select, based on the Manhattan Distance, the nearest healthy elevator. Unlike LBDR3D, First-Last adopts a distribution of those bits based on the set of virtual channels and the relative position of elevators. It is necessary to grant reachability and avoid deadlock. Also, First-Last presents a good solution for partially connected 3D-NoC using a total of 8 Virtual Chanel. It means that First-Last needs fewer virtual channels than LBDR3D. However, despite its high performance and low area overhead, First-Last does not present an online solution to tolerate runtime faults. As results, First-Last has to drop some packets during its reconfiguration phase.

To the best of our knowledge, in the literature is missing a light-weight and adaptive routing algorithm which provides good performance and runtime fault-tolerant solution for partially connected 3D-NoCs without imposing specific rules on elevator selection. Motivated by this lack in literature and in contrast to existing works described above, we propose a high performance and reconfiguration routing scheme called FL-RuNS that can tolerate manufacturing and runtime faults in the vertical links while: not imposing specific rules on elevator selection, avoiding drop packets in the presence of TSV failures, and keeping high performance of 3D-NoCs in the absence or presence of faults.

3.3 First-Last Baseline Architecture

In this section we first start discussing the partially connected 3D-NoC topology adopted in this work. Then we present a reconfigurable Elevator Bits which is used by the routing algorithm as a compass for locating healthy elevators. Finally, the First-Last routing algorithm, which is used as baseline for our runtime fault-tolerant scheme, is presented.

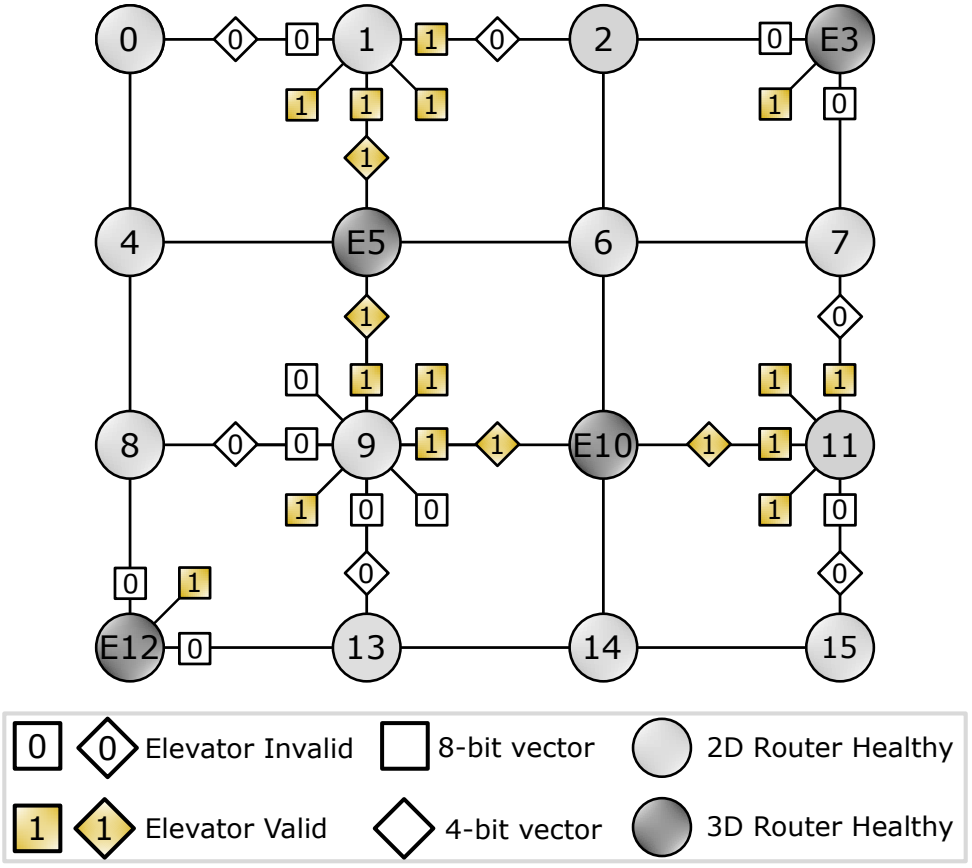

Fig. 3.2: Elevator configurable bits for initially operation with all healthy elevators.

3.3.1 3D-NoC Topology

We consider a typical on-chip router containing five major functional modules: Routing Computation, Virtual Channel Allocation, Switch Allocation, Crossbar, and Input Buffers. Also, the routers include, in addition to the usual five ports (East/**X+**, West/**X-**, North/**Y+**, South/**Y-**, Local), either an UP/**Z+** port, a Down/**Z-** port, or both (i.e. 5, 6 and 7 port configuration). The Up and Down ports of the router are connected vertically through TSVs, as shown in Figure 3.1. Each TSV pillar is connected in all layers, and we call them *elevator*. That is, each elevator connects all layers and has the Up and Down port. For more details about partially connected three-dimensional Networks-on-Chip, please check Chapter 2 Section 2.3 of this thesis.

3.3.2 Locating Healthy Elevators

In a partially connected 3D-NoC, the biggest challenge for the routing algorithm is to find a healthy elevator for delivering a packet from its source layer to its destination layer. To address this challenge, we proposed, based on our work presented in [16], a novel method for determining the possible elevators' positions using Elevator Bits inside of the routers. In this method, each router stores twelve bits that are distributed in two vectors of bits. The first vector (4-bit vector) is composed of four cardinal directions [**N**, **E**, **S**, and **W**] indicating if there is a neigh-

boring elevator one hop along the row and column. That is, if there is an elevator one hop at the **N**orth direction, the respective bit **N** in the 4-bit vector is set up to 1. The second vector (8-bit vector) represents eight cardinal directions distributed as [**N**, **E**, **S**, **W**, **NE**, **NW**, **SE**, and **SW**]. Those bits indicate the presence of at least one elevator in one direction. For example, the North or South bits are set if there is at least one elevator in the same column in the North or the South, respectively. The NE and NW, on the other hand, are used to indicate the existence of an elevator in the northeast or northwest directions, respectively.

To illustrate how these two vectors of bits are set, let us consider the example shown in Figure 3.2 where only the bits pointing to elevators are set to 1. In this example, the *router 9* knows that there are elevators at North ($E5$), East ($E10$), NE ($E3$), and SW ($E12$) directions since the corresponding elevator position bits from its 8-bit vector are set to 1: [11001001]. Also, the *router 9* knows that there are two more elevators with one hop of distance at North ($E5$) and East ($E10$) directions. In this case, the 4-bit vector is set up as follows: [1100]. Additionally, since the elevators also have the 8-bit vectors, the *elevator E3* knows that others elevators exist at SW ($E5$, $E10$, or $E12$) direction.

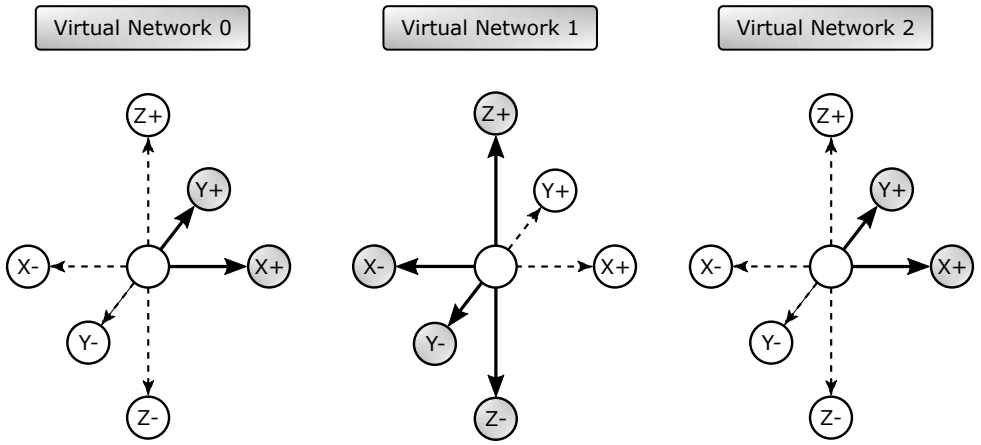

Fig. 3.3: Virtual Network decomposition for First-Last [15].

3.3.3 First-Last: The baseline algorithm

The First-Last is an adaptive routing algorithm that needs a total of eight virtual channels distributed along **X**($X0+, X0-, X1+$), **Y**($Y0+, Y0-, Y1+$), and **Z**($Z0-, Z0+$). The numbers **0** and **1** refer to virtual channels 0 and 1, respectively, and the **+** and **-** symbols represent the positive and negative channels, respectively. These channels are partitioned into three virtual networks as shown in Figure 3.3, and each virtual network has an acyclic configuration to avoid deadlock. The virtual networks *VN0* and *VN2* both use the same positive direction distributed as follows: *VN0* uses the $X0+$ and $Y0+$; *VN2* uses $X1+$ and $Y1+$. The second virtual network (*VN1*) includes the remaining directions, i.e. the negative direction ($X0-, Y0-$) as well as the

Up/Down direction ($Z0+, Z0-$). In addition to virtual network definitions, the packets must traverse virtual networks only in increasing order (*VN0* $\rightarrow$ *VN1* $\rightarrow$ *VN2*).

As shown in Figure 3.3, the First-Last routing algorithm is based on the three virtual network definition (VN0, VN1, and VN2) and its logic can be described as follows. When the destination is on the same layer as the source router, the routing algorithm selects the channels of VN0 and/or the channels of VN1 to achieve the packet's destination. On the other hand, when the destination is not on the same layer as the source router, first the routing algorithm needs to search for an elevator using either the channels of VN0 (in case the elevator is at the positive direction of the source router) or VN1 (in case the elevator is at the negative direction of the source router). Then, after reaching an elevator, the vertical channels of VN1 are used to go Up/Down, and finally, if it is necessary, the channels of VN2 are selected to reach the final packet's destination. It is worth mentioning that the First-Last uses the Manhattan distance to select the closest elevator, which means that packets are sent to its destination layer as soon as possible.

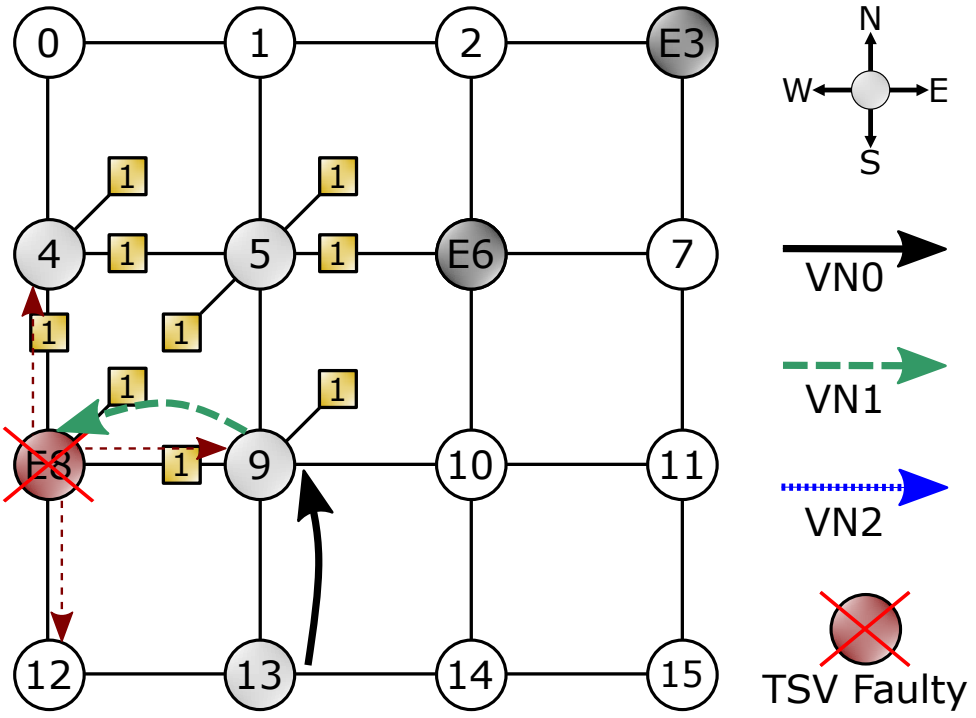

Fig. 3.4: An example where the packet cannot reach a healthy elevator using VN1. Here, the packet must be dropped since there is not another elevator at negative direction from elevator E8.

The First-Last routing algorithm can support all permanent TSV failure patterns that are known before system startup because it does not restrict the position and the number of TSV links. Also, the Elevator Bits of First-Last is configured during the startup phase, which means that only healthy elevators are selected. However, First-Last is not resilient to runtime failures. In particular, for a packet that takes the negative direction (VN1 channels), for searching for a

healthy elevator. This scenario is shown in Figure 3.4, where a packet takes *VN1* channels in the direction to elevator *E8*. But, since *E8* is a faulty elevator this packet must be dropped to avoid a deadlock condition.

In order to solve this problem, we propose a novel fault-tolerant scheme which is described in detail in Section 3.4.

3.4 FL-RuNS Routing Schemes

In this section, we will present a mechanism for propagating the elevator status and describe how the Elevator Bits (12-bit vector location) is dynamically reconfigured. Then we will define the asymmetric 1-flit-dedicated virtual channel and how it can be used as escape channels during an event of TSV failures. Finally, we will explain in detail the FL-RuNS routing algorithm procedure as well as the proof of deadlock-freedom.

3.4.1 Propagating Faulty Elevators

Providing global information about the location of faulty elevators may improve the performance of the routing algorithm. The elevator failures are propagated as shown in Figure 3.5. The TSV status notification signals are connected to all adjacent routers of the elevator's row and column allowing routers to share the status of the elevator. After detecting a TSV fault, each elevator transmits a signal to its neighboring routers signalizing that it can not work as an elevator anymore. For example, as shown in Figure 3.5, let us assume that *elevator E5* is faulty. In this case, the *elevator E5* will send its signal status to *routers 1 and 9*, in the column, and to *routers 4 and 6*, in the row, informing its faulty status.

After propagating the TSV status among the four neighboring routers closest to the elevator, the next step is to disseminate this information to all others routes. Figure 3.5 illustrates a mechanism to propagate the status of the elevator to all routers. Each router transmits three cardinal signals to signalize the location of a healthy elevator to its neighbor's router. As an example, let us consider the status propagation from routers 6 and 9 to router 10. In this case, the router 6 sends three signals [N, NE, NW] to the router 10 signalizing the presence of an elevator at North, NorthEast and/or NorthWest directions. As shown in Figure 3.5, this signal vector [N, NE, NW] of router 6 was initially [0, 1, 1] when the elevators E5 and E3 were all healthy. However, since the elevator E5 is faulty, the signal vector [N, NE, NW] of router 6 must be changed to [0, 1, 0]. It means that the elevator at West direction from router 6 is not working, but there is at least one elevator at East or NorthEast from router 6. It is worth noting that this process is repeated by the router 6 for all remaining planar ports such as North (through signals [S, SE, SW]), East (through signals [W, NW, SW]), and West (through signals [E, NE, SE]).

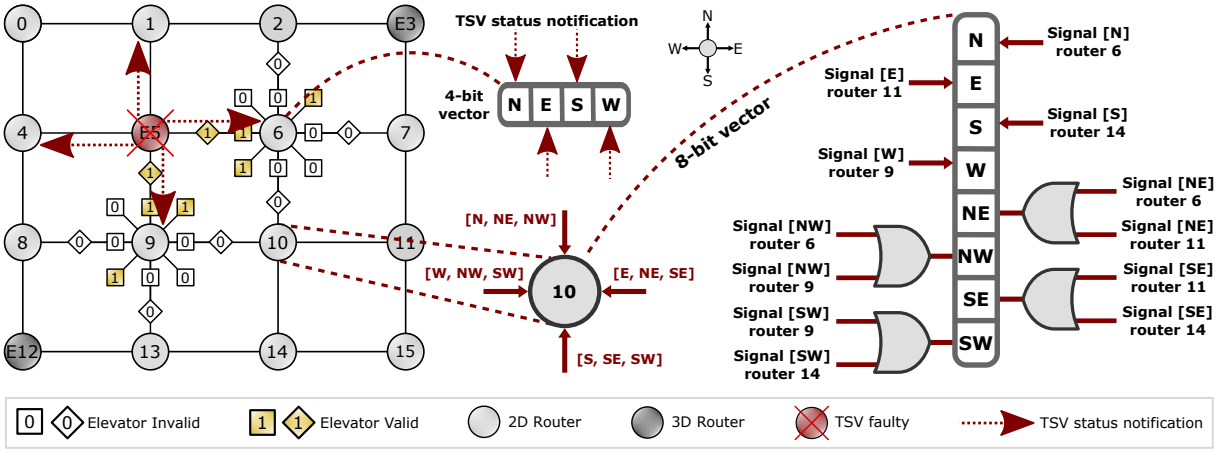

Fig. 3.5: TSV status propagation through rows and columns with the reconfiguration scheme for 4-bit and 8-bit vectors (12-bit).

All these signals are connected to the router through a combinational logic, which is used to update the 8-bit locations of the Elevator Bits. According to Figure 3.5, to set to one the bit [N] (North) of router 10, we need only the North signal from router 6 directly connected to the bit [N] of the 8-bit vector. However, to set the bit NW, we need the signal Northwest from router 6 and router 9. If one of them is equal to 1, then the bit NE will be set to 1 too. This process will be repeated until all the 8-bit vector can be updated.

3.4.2 1-Flit-Dedicated Virtual Channels

A well-known technique to avoid deadlock is the use of virtual channels distributed in acyclic virtual networks [18]. Furthermore, packets need to follow the virtual networks only in an ascendant or descendent order [41]. Although those conditions are sufficient to avoid deadlock, they can significantly increase the number of virtual channels when both deadlock-freedom and fault-tolerant conditions are taken into account. With this issue in mind, we propose an asymmetric 1-flit-dedicated virtual channel that can be used as escape channel. That is, the 1-flit-dedicated virtual channel is one 1-flit FIFO buffer which is employed exclusively as an alternative channel to overpass runtime failures in the vertical links.

Figure 3.6 shows the architecture of the FL-RuNS with one 1-flit FIFO buffer along the North, East, Up, and Down input ports. Those four asymmetric 1-flit-dedicated virtual channels were added to the baseline First-Last architecture as shown in red in Figure 3.6. In order to increase the reliability, we combine and partition those virtual channels into four virtual networks as shown in Figure 3.8. The virtual networks *VN0* and *VN1* are the same as that of the First-Last. However, the virtual network *VN2* includes two 1-flit-dedicated virtual channels along $Z(Z1+, Z1-)$. Furthermore, the virtual network *VN3* contains only the 1-flit-dedicated virtual channels along $X(X1-)$ and $Y(Y1-)$. In summary, these four 1-flit-dedicated virtual channels are used only as escape channels for the packet that does not find a healthy eleva-

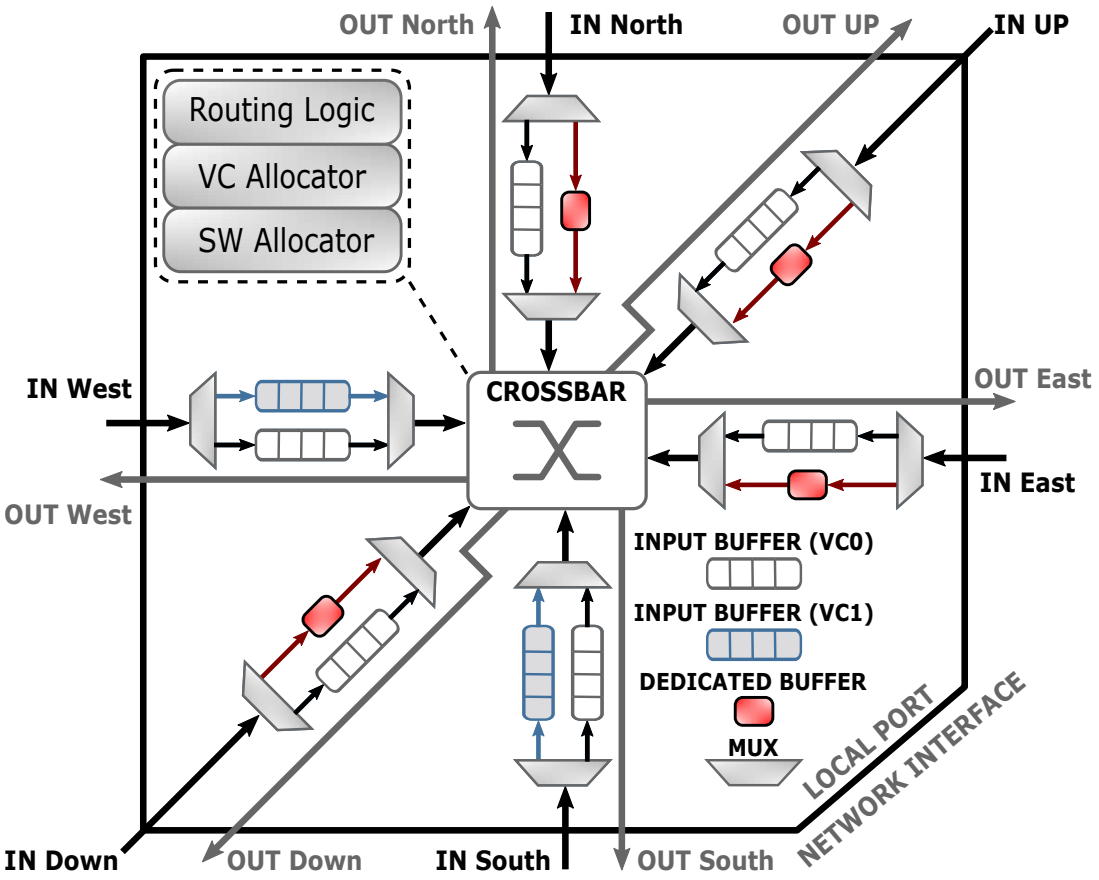

Fig. 3.6: Architecture of FL-RuNS with 1-flit-dedicated virtual channel. The 1-flit-dedicated virtual channel (1-flit Fifo Buffer) is used as alternative virtual channels after a runtime failures in vertical connecxion.

tor in the *VN1* but needs to take the *VN2* then *VN3* to search again for a healthy elevator. In order to avoid deadlock, the packets must traverse virtual networks only in increasing order ($VN0 \rightarrow VN1 \rightarrow VN2 \rightarrow VN3$).

The idea here is to provide an asymmetric distribution of the virtual channels to achieve both fault-tolerance and deadlock-freedom while keeping the hardware cost reasonable. Using 1-flit-dedicated virtual channels, we can rearrange the router in a way that it is possible to provide full flexibility for choosing the best place to plant the TSVs as well as low-weight alternative for tolerating runtime failure in the vertical connection. Even though these 1-flit-dedicated virtual channels increase hardware overhead of the router, we still present here a better solution than the one of our previous work [24] where all the flits of a packet must be stored into an escape-buffer to search for an elevator after a runtime fault. In other words, the escape-buffer must be as large as the size of the packet, which means that the area overhead will increase with the size of the packet. However, unlike our previous work, the 1-flit-dedicated virtual channels do not increase as the size of the packet increases. It means that the area overhead will be fixed in one 1-Flit FIFO buffer independently of the number of flits into the packet. Also, comparing the Figure 3.6 and Figure 3.7, it is possible to see that the total number of buffers employed by

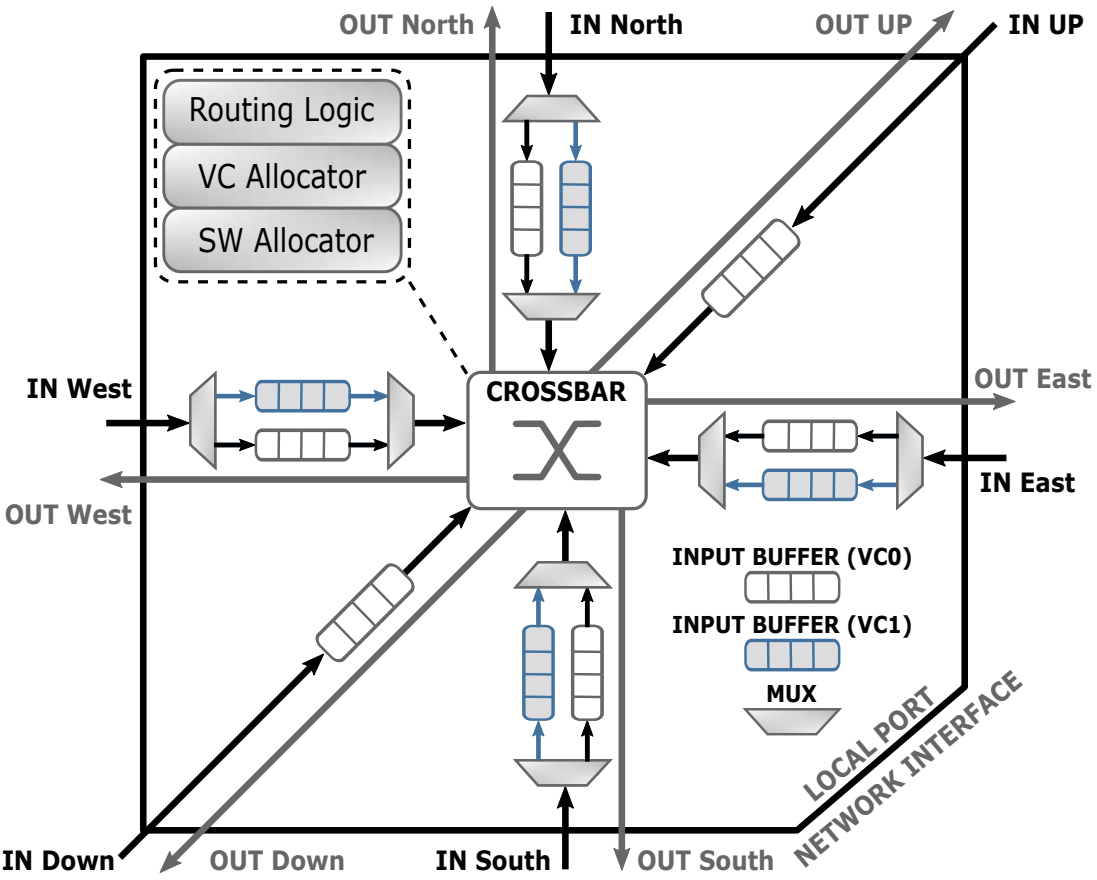

Fig. 3.7: Elevator-First architecture with distributed buffers. All its buffers are used only as a mechanism to avoid deadlock.

the FL-RuNS is still lower than Elevator-First. While the Elevator-First needs two completely symmetric buffers in all planar directions, the FL-RuNS uses only two complete buffers at West and South, and two 1-flit-dedicated buffer at North and East. In other words, our solution still uses less area and power than Elevator-First. A detailed hardware comparison will be presented in Section 3.5.3.

3.4.3 Proposed Routing Algorithm

The FL-RuNS routing algorithm needs the same number of virtual channels as First-Last. Also, it needs more four 1-Flit-Dedicated Virtual Channels, as shown in Figure 3.8. The pseudo algorithm is presented in Algorithm 1. As an input, the algorithm takes a bit vector $Dest$ describing the router destination position, a bit vector Cur describing the current router position, a 4-bit and 8-bit vector containing the elevators' directions (elevators bits), and the current virtual network number v_in. The routing algorithm results are an output port $Direction$ as well as a new virtual network number v_out if moving to the next virtual network is necessary. According to Algorithm 1, the logic of the routing algorithm based on the four virtual network definitions is described as follows::

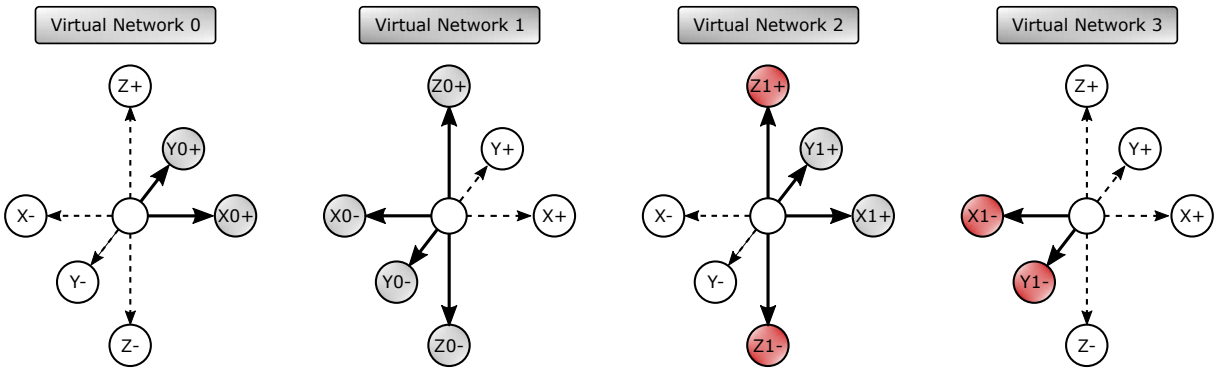

Fig. 3.8: Decomposition of Virtual Network for FL-RuNS. The 1-Flit-Dedicated virtual channels are represented in red: Z1+, Z1-, X1-, and Y1-.

3.4.3.0.1 Source and Destination are on the same layer If the destination is on the same layer as the current router (Algorithm 1 - Lines 27 to 33), routing is performed adaptively following the positive directions $\{North, East\}$ then negative directions $\{South, West\}$. The selection between two possible directions is made based on the congestion values (i.e., the free buffer slot in the neighboring routers).

3.4.3.0.2 Source and Destination are not on the same layer The packet needs to be forwarded to a healthy elevator in the source layer, transferred to the destination layer, and delivered from the elevator to the final destination. If the current router is an elevator (Algorithm 1 - Lines 2 to 5), then the packet is forwarded appropriately either to the up or down port. Moving up and down is possible in both *VN1* and *VN2*, so the *VN* number must be updated. If the current router is not an elevator (Algorithm 1 - Lines 6 to 25), the packet is adaptively routed towards the selected elevator, based on the 4 and 8-bit vectors, following the positive and negative directions.

Figure 3.9 illustrates two examples of FL-RuNS routing operation. In the first example, the source node $S1$ delivers packets to the destination at node $D1$ using the fault-free elevator $E1$. First, the routing algorithm checks the 4-bit elevator position to verify if there is a neighboring elevator with one hop distance. Since there is not a one-hop elevator, the routing algorithm then checks the 8-bit elevator position and sends the packet toward North (router $H1$) since there is an elevator at Northeast $E1$ and another one at North (elevator $E2$). The channels of the *VN0* are used to forward the packet toward router $H1$. When the packet arrives at router $H1$, a decision needs to be made between continuing to North (elevator $E2$) or changing to East (elevator $E1$). This decision is made when the routing algorithm checks the 4-bit vectors and verifies that an elevator is available at one hop East. Then, the packet is sent to elevator $E1$ using the same *VN0*. In the elevator $E1$, the channels of *VN1* are applied to deliver the packet to its destination layer (*UP*). Finally, the packet is delivered to its router destination by using the channels of *VN1*, toward $H2$, and *VN2*, toward $D1$.

Algorithme 1 FL-RuNS Routing Algorithm Procedure

Require:
$Dest$: Destination router position bits
Cur: Current router position bits
$Elev$: Elevator location bits (12-bit)
v_in: Current virtual network

Ensure:
$Direction$: Output port
v_out : Output virtual network

if $Cur.Z \neq Dest.Z$ **then**
  if $Cur.isElevator$ **then**
    $Direction = (Cur.Z < Dest.Z \; ? \; Up : Down)$
    $v\_out \leftarrow V$ ▷ To go Up/Dn must be **VN1** or **VN2**
  else
    $v\_out \leftarrow v\_in$
    if $v\_out == 0$ or $v\_out == 2$ **then**
      ▷ Need to search for elevator in **VN0** or **VN2**
      if $Elev.NE \; || \; .NW \; || \; .SE \; || \; .N \; || \; .E$ **then**
        $Based\ on\ the\ 4bits\ and\ 8bits\ vectors.$
        $Direction = \{North, East\}$
      end if
    else
      $v\_out \leftarrow 1$ ▷ Search for elevator in **VN1**
      if $Elev.SW \; || \; .NW \; || \; .SE \; || \; .W \; || \; .S$ **then**
        $Based\ on\ the\ 4bits\ and\ 8bits\ vectors.$
        $Direction = \{South, West\}$
      else
        $v\_out \leftarrow 2$ ▷ Search for elevators in **VN2**.
        if $Elev.NE \; || \; .N \; || \; .E$ **then**
          $Direction = \{North, East\}$
        end if
      end if
    end if
  end if
else
  if $Dest.X < Cur.X \; || \; Dest.Y < Cur.Y)$ **then**
    ▷ Taking positive directions using **VN0** or **VN2**
    $Direction = Adaptive\_North\_East(Dest)$
  else
    ▷ Taking negative directions using **VN1** or **VN3**
    $Direction = Adaptive\_South\_West(Dest)$
  end if
end if

In the second example, the source node $S2$ targets destination at node $D2$ through elevator $E3$. First, the routing algorithm checks the Elevator Bits (4-bit and 8-bit vectors location) and

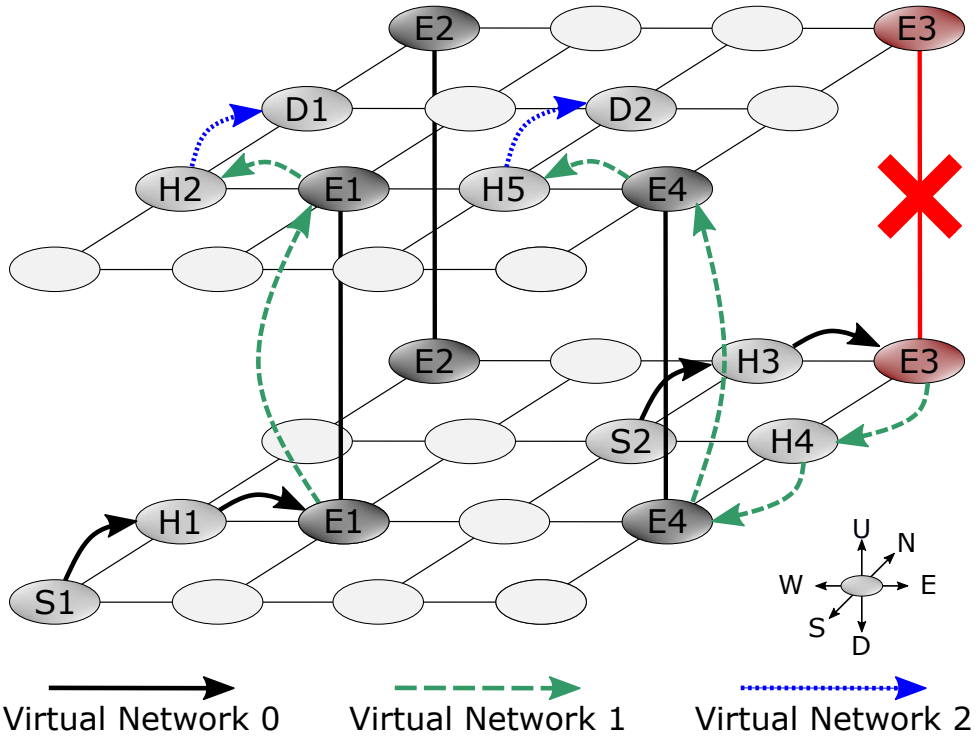

Fig. 3.9: Example of FL-RuNS in a scenario with and without TSV failure

sends the packet to North toward $H3$. When the packet arrives at the router $H3$, the routing algorithm checks the Elevator Bits and sends the packet at East toward $E3$. However, since the packet arrives at elevator $E3$ before the router $H3$ can be notified about one fault in $E3$ through the TSV status propagation, the solution here is rerouting this packet to another elevator. The mechanism adopted in this Chapter to recalculate the routing computation is based on the one proposed by [25]. So, after performing a new routing computation, the elevator $E3$ searches for another one using the 8-bit vector and sends the packet toward the elevator $E4$ using negative direction (*VN1*) until reaching the router $H5$. Finally, the packet is changed to the positive direction (*VN2*) in the router $H5$ to arrive at its final destination $D2$.

Figure 3.10 illustrates the worst failure example that FL-RuNS can tolerate using the 1-flit-dedicated virtual channels. In this example, the source *router S1* wants to send a packet to a *router D1*. The packet is routed to North and arrives to the *router H1*. Then the packet is routed to West toward *elevator E1* after its virtual channel is changed to *VN1*. However, if faults occur in the elevator $E1$ when the packet arrives, the routing algorithm must search for another elevator using the current *VN1*. Since in this example there are no more elevators at West or South directions relative to the position of elevator $E1$, the routing algorithm then changes the virtual network from *VN1* to *VN2* and searches for an elevator at the positive direction using *VN2*. In this case, the packet is routed to North toward elevator $E2$. However, to go Up/Down, the packet has to take the 1-flit-dedicated VC available in the *VN2*. After the packet arrives at its destination layer, it then takes the East (positive direction) using *VN2* toward the router $H4$.

Finally, the packet is routed to South (negative direction) toward its destination router $D1$ using again the 1-flit-dedicated VC available into *VN3*.

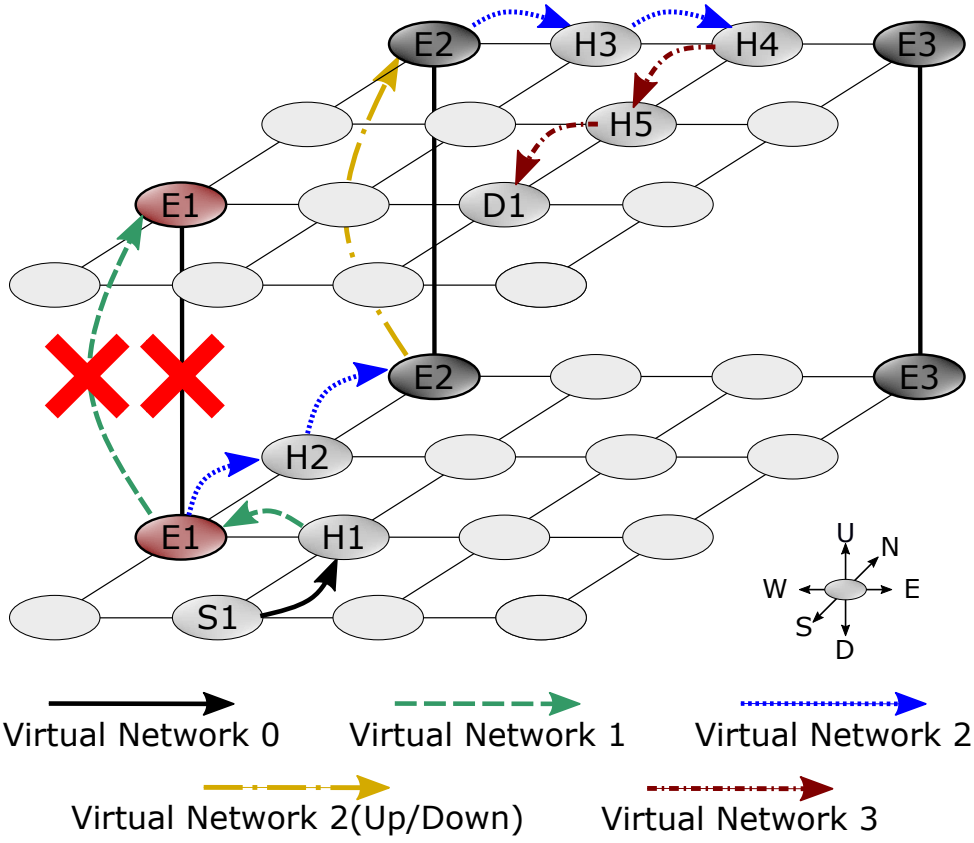

Fig. 3.10: Example of FL-RuNS using the 1-flit-dedicated virtual channel to rerouting packets toward a healthy elevator.

Figure 3.10 illustrates the worst failure example that FL-RuNS can tolerate using the 1-flit-dedicated virtual channels. In this example, the source *router S1* wants to send a packet to a *router D1*. The packet is routed to North and arrives to the *router H1*. Then the packet is routed to West toward *elevator E1* after its virtual channel is changed to *VN1*. However, if faults occur in the elevator $E1$ when the packet arrives, the routing algorithm must search for another elevator using the current *VN1*. Since in this example there are no more elevators at West or South directions relative to the position of elevator $E1$, the routing algorithm then changes the virtual network from *VN1* to *VN2* and searches for an elevator at the positive direction using *VN2*. In this case, the packet is routed to North toward elevator $E2$. However, to go Up/Down, the packet has to take the 1-flit-dedicated VC available in the *VN2*. After the packet arrives at its destination layer, it then takes the East (positive direction) using *VN2* toward the router $H4$. Finally, the packet is routed to South (negative direction) toward its destination router $D1$ using again the 1-flit-dedicated VC available into *VN3*.

FL-RuNS can guarantee 100% of packets delivery under the most known failures scenarios in the 3D-NoC architecture, such as the examples illustrated above. However, there are scenar-

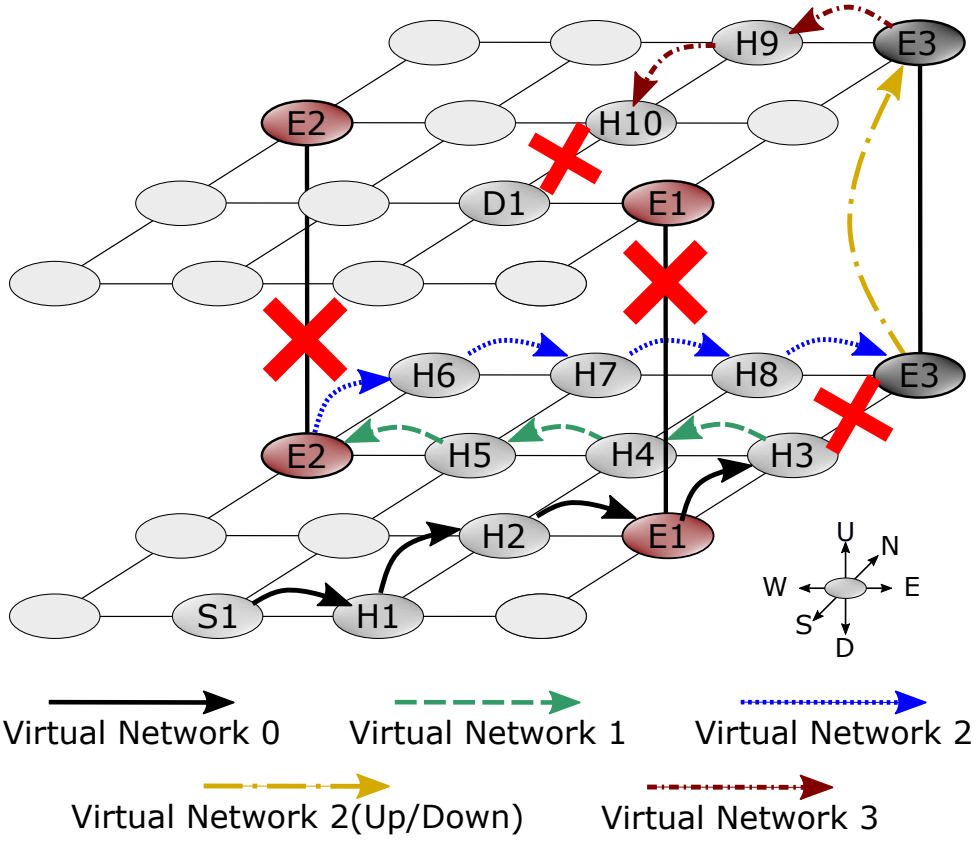

Fig. 3.11: Example of FL-RuNS in a fault scenario which cannot guarantee packet delivery.

ios with multiple and simultaneous failures in both 2D-routers and 3D-routers (elevators) which FL-RuNS cannot support. Let us imagine an example shown in Figure 3.11 where a packet initially searches for a healthy elevator *E1* in a positive direction using *VNO*. This packet cannot find a healthy elevator using *VNO* because there is a failure in the elevator *E1* and in the output port *North* of the 2D-router *H3* toward the elevator *E3*. So, the packet must change its direction and search for elevator*E2* in the negative direction using the virtual channel *VN1* and the 2D-routers *H4* and *H5*. But again when this packet arrives at elevator *E2* a failure happens and it cannot go to its destination layer. In this case, this packet must change its direction to search for another healthy elevator using the virtual channel *VN2* (positive direction). Assuming that this packet found the healthy elevator *E3* and arrived at its destination layer through the 1-flit-dedicated virtual channel Up/Down, it then must go toward the Southwest direction to reach its final destination *D1*. So, the packet must change its virtual network to *VN3* in order to go to Southwest direction. However, if there is a failure in a 2D-router on the Southwest path, this packet may not bypass this failure node and must be dropped. That was exactly what happened in the example of Figure 3.11 due to a failure in the output port *South* of the 2D-router *H10*. So, under this fault scenario, the packet must be dropped to avoid deadlock or livelock.

A fault scenario like that one shown in Figure 3.11 may not occur due to the high number of multiples and simultaneous failures (two elevators and two 2D-routes), but it should be

considered. Although FL-RuNS can tolerate some fault scenarios in 2D-routers, it was initially designed to tolerate faults on vertical connections, since vertical connections are more sensitive to permanent and transient faults than horizontal ones. As demonstrated in Section 3.5, FL-RuNS shows better adaptability and reliability than the state-of-the-art 3D routing algorithms since it can dynamically select positive as well as negative directions in the planar directions. Algorithms such as Elevator-First [39] and CoBRA [102], for example, take the *"X"* direction before the *"Y"* direction to avoid deadlock. This lack of adaptability in the *"X"* directions can limit the algorithms' reliability, which makes it difficult for them to support failures simultaneously in the vertical and horizontal connections.

3.4.4 Deadlock-freedom

A cyclic dependency occurs when positive and negative directions have to be taken along at least two dimensions. Consequently, in order to avoid deadlock, a routing algorithm must divide its channels into disjoint partitions and the transitions between partitions must be allowed only in a consecutive order [41]. In this case, the proposed fault-tolerant routing scheme is deadlock-free because the packets traverse virtual networks only in increasing order ($VN0 \rightarrow VN1 \rightarrow VN2 \rightarrow VN3$). Also, none of the defined virtual networks spans two full dimensions.

Furthermore, a deadlock is a situation in which packets are waiting for each other to release the reserved channels and are unable to make progress. In particular, if a waiting activity never finishes, it implies that the deadlock situation will persist forever. So, in order to prevent deadlock, the 1-Flit-dedicated virtual channels are used only in case of TSVs failures. It means that packets in transit to a faulty elevator are not blocking reserved resources since those packets can use the 1-flit-dedicated virtual channel as an escape path to search for a healthy elevator at positive direction.

3.5 Simulation Results and Discussion

We implemented and evaluated our proposed routing algorithm using SystemVerilog based cycle-accurate 3D mesh NoC simulator, which was created by extending the open source 2D-NoC Netmaker library [77] by adding support for a 3D network. This library was used to implement the FL-RuNS described in this Chapter, the Elevator-First described in [39], the First-Last presented in [15], the CoBRA presented in [102], and also our previous version named RuNS described in [24]. Two configurations of 3D-mesh NoC were considered for the experiments. In the first configuration, we start the experiments with a 4x4x4 mesh to analyze its behaviour under a density of vertical link of 25% (i.e., four elevators). In the second configuration, we adopted a configuration of mesh with 256 Nodes distributed in a 8x8x4 mesh and a vertical link density of 12.5% (i.e. eight elevator). The idea here is the analysis of the performance of our

method using a large 3D-mesh network. All the routers have 4-flit FIFO, a packet size of 5 flits, and perform virtual channel allocator followed by switch allocator.

Simulations were done to evaluate the reliability of FL-RuNS taking into account permanent and transient faults in the elevators. We have adopted transient faults as faults that temporarily disable the functionality of the 3D-router during its runtime operation. This means that transient faults convert a 3D-router into a 2D-router for a random period of time during the runtime phases. On the other hand, permanent faults are failures that disable the router at the begining of startup phase. In other words, a 3D-NoC starts with fewer vertical links than the ones expected during its manufacturing phase. So, a faulty TSV was characterized by simply disabling the function of the Elevator and by propagating the TSVs statuses, as illustrated in Figure 3.5. As described in the Section 3.4.1, the propagation of TSVs status is necessary to inform the neighboring routers about failures in the elevators. In summary, FL-RuNS can fully configure the NoC by just changing its routing algorithm decision, which can be done online for transient faults, or even off-line for permanent faults.

3.5.1 Performance and reliability analysis under a 4x4x4 mesh

In order to evaluate the average latency and the reliability, we consider a 4x4x4 mesh 3D-NoC with four Elevators ($E3, E5, E10,$ and $E12$) as shown in Figure 3.2. Additionally, three different synthetic traffic patterns were used: Uniform random, Bit-Complement, and Shuffle.

First, we begin evaluating the average latency for each traffic pattern running 100000 cycles simulation without fault injection. Figure 3.12 shows the latency comparison for Elevator-First, CoBRA, First-Last, RuNS, and FL-RuNS routing algorithms when all TSVs are healthy. In all cases, Elevator-First and First-Last provide slightly better performance than FL-RuNS. This can be attributed to both Elevator-First and First-Last selecting the closest TSVs during the manufacturing process and off-line configuration respectively. Also, FL-RuNS starts searching for a healthy elevator at positive direction before searching for a healthy one at negative direction. Additionally, Elevator-First uses two more virtual channels than FL-RuNS which increases its general performance. On the other hand, FL-RuNS shows better performance results than CoBRA because it can adaptively take the less congested path while CoBRA has to take the East direction before moving toward the West, which increases congestion in X dimension. Also, FL-RuNS shows better performance results than RuNS due to the additional virtual network, the better mechanism for propagating TSV status, more adaptivity to the routing algorithm, and the additional 4-bits in the Elevator Bits.

Second, we evaluate the average latency with single and double faults, as shown in Figure 3.13 and 3.14 respectively. As expected, the performance of FL-RuNS is slightly reduced in the presence of faults. This degradation in performance is noticed, firstly because some packets must take the 1-flit-dedicated virtual channel to search for a healthy elevator, and secondly because the status of the elevator fault, as well as the reconfiguration of the entire router, take a

few clock cycles to be propagated.

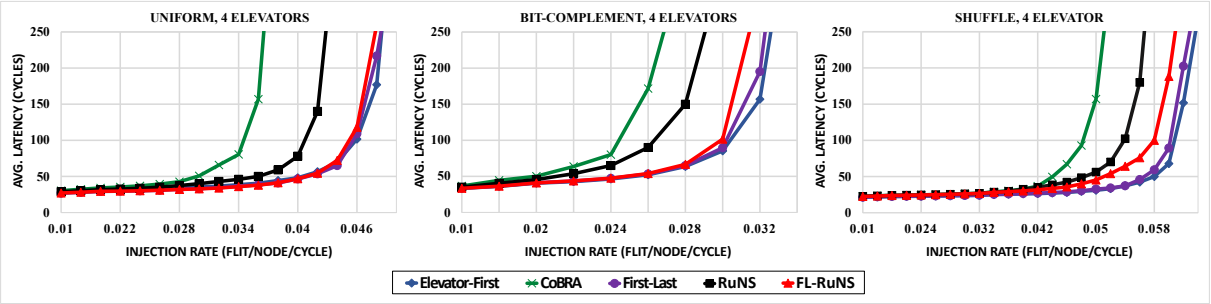

Fig. 3.12: Average packet latency for an 4x4x4 NoC without fault injection.

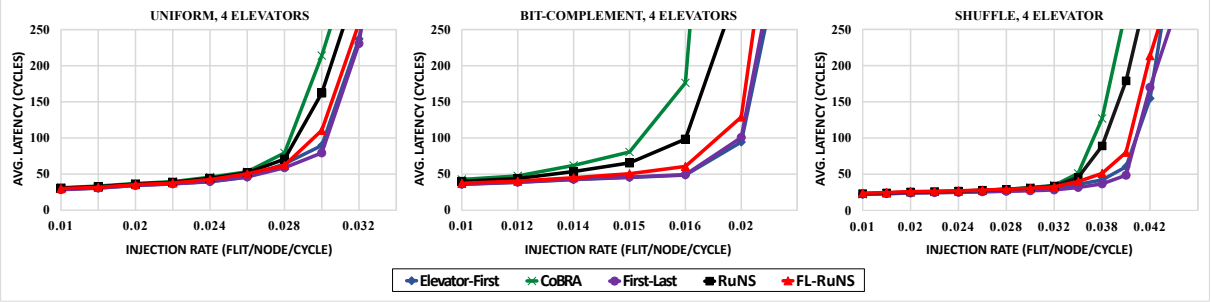

Fig. 3.13: Average packet latency for an 4x4x4 NoC with single fault injection.

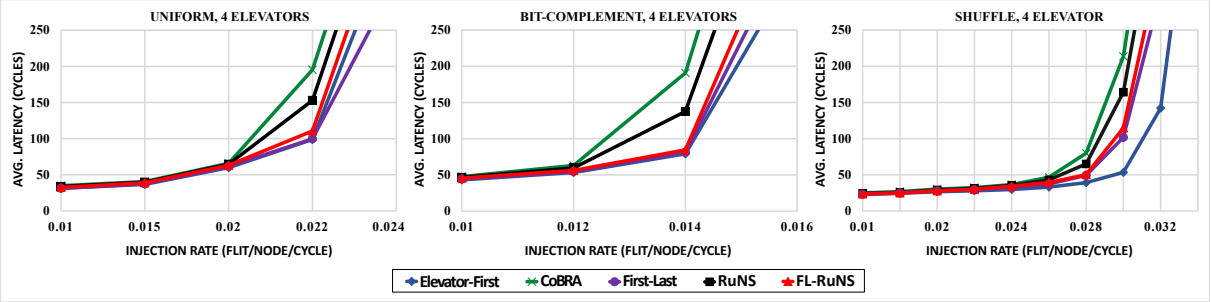

Fig. 3.14: Average packet latency for an 4x4x4 NoC with double fault injection.

Finally, for the reliability comparison, each simulation injects a fixed number of flits (10000/core) to enable comparisons for successful arrival rates. In other words, the measure of the reliability defined in this Chapter is the percentage of flits successfully delivered to the target destination. Figures 3.15 and 3.16 show the normalized reliability comparison for the Elevator-First, CoBRA, First-Last, RuNS, and FL-RuNS under the effect of single and double faults respectively. We can observe that for single faults the reliability of CoBRA is almost the same as FL-RuNS. However, CoBRA lost packets when failures occurred in the eastmost elevator $E3$ (see Figure 3.2). This occurred because some packets were dropped in the reconfiguration phase until the CoBRA returns to its stable condition. The analysis of double faults shows that FL-RuNS performs better than CoBRA since FL-RuNS can send all its packets to a healthy elevator, while

CoBRA needs at least one elevator placed in the eastmost or westmost column in order to avoid dropping packets. FL-RuNS and RuNS show the same reliability for single and double faults since both share a similar escape buffer idea, which was inspired by the virtual source concept.

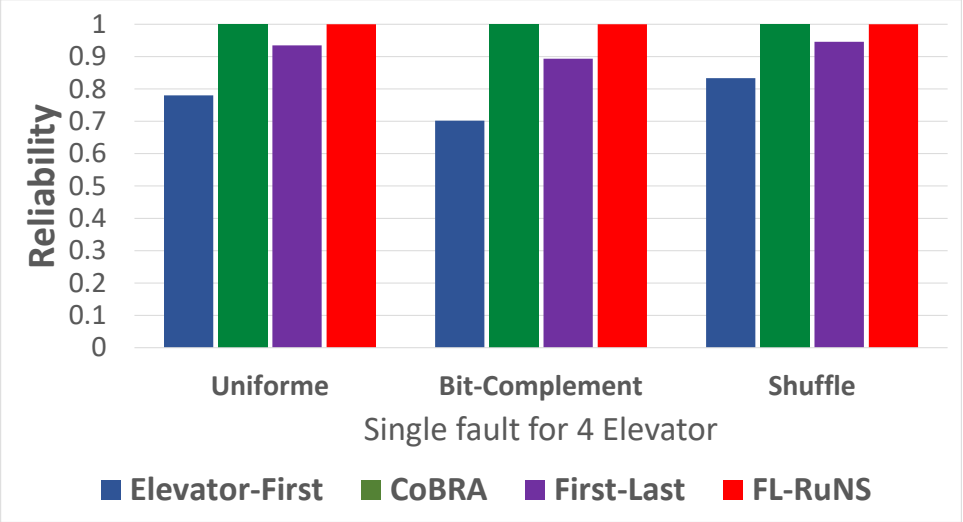

Fig. 3.15: Reliability under single fault for 4 TSVs

Fig. 3.16: Reliability under double faults for 4 TSVs

Comparing both First-Last and Elevator-First with FL-RuNS, Figures 3.15 and 3.16 demonstrate that FL-RuNS shows better reliability than either of those algorithms. Although First-Last shows slight better latency performance than FL-RuNS, mainly because First-Last uses a mechanism to select the closest elevator based on Manhattan distance, the reliability performance of First-Last is lower than FL-RuNS. So First-Last shows reliability up to 20% less than FL-RuNS. The Elevator-First shows the worst scenario for faults because it cannot adapt itself to faults at runtime. Furthermore, Elevator-First does not have any off-line mechanism to reconfigure the

elevator selection after the manufacturing process. Here again, the two additional virtual channels into the Elevator-First boost its latency performance in comparison to FL-RuNS. However, Elevator-First shows reliability up to 35% less than FL-RuNS.

3.5.2 Performance and reliability analysis under a 8x8x4 mesh

In order to analyze FL-RuNS under a large 3D-NoC, we have performed two simulations (with and without vertical failures) using an 8x8x4 mesh architecture. Also, we have adopted a very low density of vertical connection to investigate the performance of the FL-RuNS under a congestion situation. To do it, only eight elevators are placed in the network, which means a density of 12.5%. The performance metric we consider is the average packet latency, which is the average elapsed time between the queuing of a packet in the network interface and the reception of its tail flit at the destination network interface. Here again, the synthetic traffic patterns used in those two simulations include uniform, bit-complement, and shuffle. The average latencies are plotted in Figures 3.17 and 3.18.

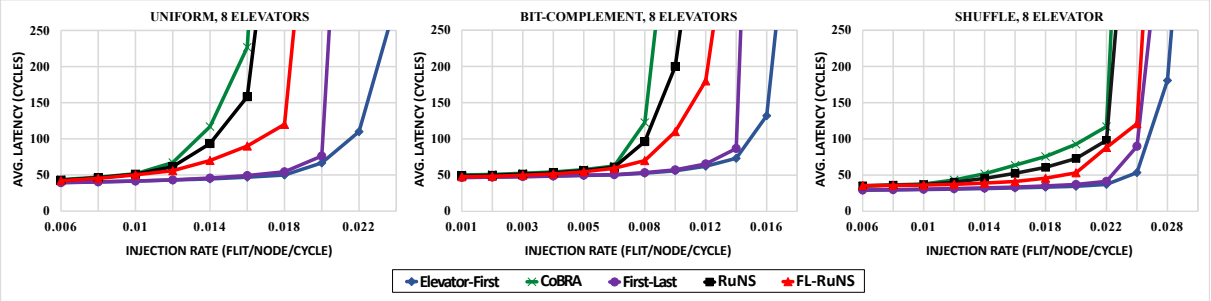

Fig. 3.17: Average packet latency for an 8x8x4 NoC without fault injection.

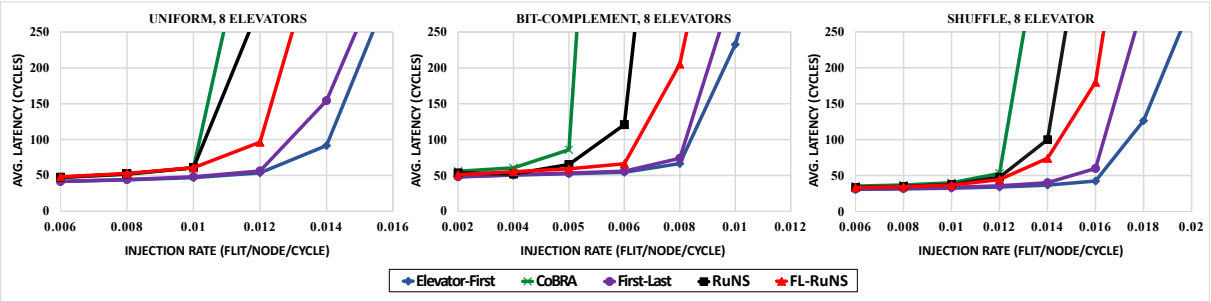

Fig. 3.18: Average packet latency for an 8x8x4 NoC with double fault injection.

Figure 3.17 shows the performance results for a simulation without fault injection. It shows that if the number of TSV increases, the performance of the FL-RuNS increases proportionally. Like in the 4x4x4 simulations, FL-RuNS shows better performance than CoBRA and RuNS for all three synthetic traffics as well as a slight slower performance when compared with Elevator-First and First-Last. However, in shuffle traffic, where many nodes communicate within the

same layer, we see that the FL-RuNS maintains a performance close to First-Last and Elevator-First.

Figure 3.18 shows the performance results for a simulation with double fault injection. We can see that most of the results are similar to those shown in Figure 3.17. However, because of the low density of vertical connections and the high simultaneous failures, the routing algorithm needs to use more the 1-flit-dedicated virtual channel to guarantee that packets can reach their destination instead of just dropping them. It can explain the degradation in the performance of FL-RuNS with double faults when compared with the performance of no faults simulation shown in Figure 3.17. Despite the performance degradation, both FL-RuNS and RuNS guarantee that all packets were delivered during the simulation. On the other hand, CoBRA dropped only a few packets during its reconfiguration phase. Because Elevator-First and First-Last do not have a mechanism to reconfigure themselves after an elevator failure event, they dropped packets during the entire runtime phase to avoid deadlock.

3.5.3 Hardware synthesis analysis

To evaluate the area and power consumption, we synthesized the Elevator-First, CoBRA, First-Last, RuNS, and FL-RuNS using Synopsys Design Compiler. The designs were setup to work with an operating frequency of 1GHz, a power supply of 1V, and a commercial STMicroelectronics FD-SOI $28nm$ library. The resulting area and power estimates for each router are summarized in Tables 3.1 and 3.2. The three types of routers previously described in subsection 3.3.1 were considered: 5-port 2D routers, 6-port 3D routers with one vertical connection, and 7-port 3D routers with both Up and Down vertical connections. It is important to note that each router was synthesized with all router's logics such as Input port, Virtual Channel control, Switch Allocator, Crossbar, and the routing algorithm. Also, all routers are configured with a flit size of 64 bits, a FIFO buffer with capacity for five flits, and a number of virtual channels inherent in each routing algorithm. The routing algorithms were implemented following the original specification, which can be found in their respective references. And finally, the Synopsys Design Compiler was configured to obtain the better results for area.

Tables 3.1 and 3.2 show that the area and power for CoBRA and First-Last are nearly the

Table 3.1: Area synthesis results

Type	**Area** (μm^2)				
(# Ports)	Elevator-First	CoBRA	First-Last	RuNS	FL-RuNS
5-Port	16302	14629	14654	17667	15308
6-Port	20374	18649	18584	21713	19561
7-Port	25107	23428	23366	26568	24315

Table 3.2: Power synthesis results

Type (# Ports)	**Power (mW)**				
	Elevator-First	CoBRA	First-Last	RuNS	FL-RuNS
5-Port	13.85	12.04	12.18	15.11	13.26
6-Port	18.82	14.90	15.07	18.06	16.63
7-Port	20.15	18.29	18.51	21.61	20.04

same for 5, 6, and 7 ports configuration. However, concerning the worst case in terms of area and power overhead, FL-RuNS shows approximately 5.2% more area and 10.3% more power than both CoBRA and First-Last. On the other hand, FL-RuNS shows approximately up to 6.1% and 11.6% less area and power than Elevator-First, respectively. Also, FL-RuNS shows a reduction of approximately 10% in area and 9% in power when compared with RuNS.

The increase in area and power can be mainly attributed to the fact that FL-RuNS needs four 1-flit-dedicated virtual channels distributed at Z1+ (Up), Z1- (Down), X1+ (West), and Y1- (South) directions. Also, FL-RunS uses a fault propagation scheme and a rerouting mechanism to recalculate and select new paths after faults. It is worth mentioning that First-Last, CoBRA, RuNS, and FL-RuNS use the same total number of virtual channels, which means that the differences of area and power are provided mainly by the routing algorithm and the fault-tolerant techniques adopted by FL-RuNS. On the other hand, to avoid deadlock, Elevator-First needs two more virtual channels than the others algorithms which can justify the lower area and power overhead between FL-RuNS and Elevator-First.

3.6 Conclusion

In this Chapter, we have presented FL-RuNS, a fault-tolerant routing scheme for partially connected 3D-NoC. The main contribution of FL-RuNS is to provide a scheme able to tolerate faults in the TSV during the manufacturing and runtime phases. Also, we have demonstrated that using an asymmetric topology configuration for virtual channels and virtual network, FL-RuNS can increase its reliability while maintaining deadlock-freedom. Our simulations show that FL-RuNS is slightly inferior to the Elevator-first and First-Last algorithm in terms of latency. On the other hand, FL-RuNS is significantly more resilient to runtime failures in the vertical connections than both Elevator-first and First-Last. The simulation results indicate that FL-RuNS guarantees 100% packet delivery under extreme scenarios for runtime and permanent vertical link failures. Additionally, FL-RuNS can be entirely reconfigured and does not impose any restriction on the position of the TSVs. Those characteristics give designers the flexibility to choose the best location for the elevator that can increase the performance of the target appli-

cation. Although we have implemented our fault-tolerant scheme over the baseline First-Last routing algorithm, this same scheme can be applied without much effort to another routing algorithm such as Elevator-First. However, the cost to adapt the Elevator-First routing algorithm and implement both the asymmetric-dedicated escape virtual channels as well as the reconfiguration mechanism can significantly increase the area and power overhead. The hardware complexity has demonstrated that FL-RuNS shows a small overhead in terms of area cost (5.2%) and power consumption (10.3%) when compared with the First-Last. However, this overhead is acceptable because FL-RuNS is still functional at high fault rates while state-of-the-art partially-vertically-connected 3D-NoC fail to deliver packets.

Part III

TOOLS FOR FAULT INJECTION IN HDL DESIGN

Chapter 4

NETFI-2: A Framework to Fault Injection in HDL-Based Design

4.1 Introduction

Over the last years, the semiconductor industry has been particularly interested in the effects of radiation and the mitigation strategies on integrated circuits such as Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs) and embedded systems in general [9]. The rationale behind this motivation lies not only in the use of these systems in harsh radiation environments [37] but also in the increasing degree of integration of devices embedded in the same chip. Recent studies have shown that the smaller the feature sizes, the greater the sensitivity to radiation-induced errors [61]. As a consequence, modern embedded systems may be susceptible to low-energy particles including those observed within the Earth's atmosphere even at ground level.

The impact of energetic particles on integrated circuits can cause alterations in the behavior of microelectronic components. These errors are known as Single Event Effects (SEEs) and can be of different types. Among them, those that result in the change of a bit of information in a register or memory cell are called Single Event Upsets (SEUs), while the transient pulses that modify the combinatorial logic are known as Single Event Transients (SETs). The greater the scale of integration, the higher probability of occurrences of transient faults, which can be a challenge to traditional fault tolerant techniques. In this context there is an increasing need to estimate the sensitivity to SEE of modern integrated circuits.

In order to study the effects of SEUs and SETs on digital circuits, tests are usually carried out under radiation beams to analyze the behavior of the device under a large particle flow [31]. However, these campaigns are very costly, they are based on the physical implementation of the Circuit Under Test (CUT) and require considerable technical and programmatic effort [117]. Consequently, *simulation* and *emulation* methodologies are increasingly popular alternatives to evaluate and predict the behavior of these circuits before manufacturing. In particular, NETlist

Fault Injector (NETFI) was proposed in [73] and extended in [74] as a method to inject faults at the Register-Transfer Level (RTL). From an user perspective, Hardware Description Language (HDL) can be conveniently provided as input while left unchanged during the fault-injection process. In general, NETFI results are particularly attractive since it combines a good controllability and observability of the experiment with the ability to inject faults in a single clock cycle. Nonetheless, this methodology has been criticized for:

a) The complexity and rigidity associated with the controller responsible for the execution of the fault injection in the FPGA [105].

b) The lack of accuracy on the SET estimation when considering implementations for ASICs based on simpler gates than those use in the FPGA [73].

In this thesis, NETFI-2 is introduced as an evolution of the NETFI methodology that tackles the aforementioned weaknesses. Unlike its previous version, NETFI-2 allows controlling the fault-injection campaign from the same FPGA where the CUT is instantiated, thus minimizing the amount of hardware required. NETFI-2 is based on a controller implemented on an embedded MicroBlaze processor, which allows a straightforward design of extensive fault-injection campaigns while facilitating the identification of sensible circuit elements. Furthermore, to better appraise the final CUT behavior, the SET injection can be tailored by specifying the size of the Look-Up Tables (LUTs) of the target FPGA that will be used for the implementation of the combinational part of the CUT. This is of particular interest for evaluating designs that will be implemented in ASICs with smaller gate sizes. In order to demonstrate these advantages and to validate the methodology, we have used the circuits of a Bayesian Machine [47] and a Suport Vector Machine [116] as case studies. Also, a radiation test in a Suport Vector Machine was performed to validate the correlation between the fault injection campaign using FPGAs and the real radiation test.

4.2 State-of-the-art

Emulation-based and simulation-based fault injection are two widely adopted methods to test and analyze the effects of radiation in electronic circuits. In simulation-based fault injection, the CUT is simulated by altering its logical values during the fault-injection campaign. High observability and controllability of all the modeled components are among the most relevant advantages of this technique [8]. However, simulation-based fault injection approaches require significant computational effort since they just simulate the execution of the circuit at behavioral and structural level [66, 92, 107, 115]. As a result, their capability for analyzing a large number of faults on circuits with millions of gates is very limited. Thus, emulation-based fault injection schemes have emerged as an alternative to accelerate the fault injection campaigns experiments.

By exploiting reconfigurable devices (i.e., FPGAs), emulation strategies also consider time constraints providing a better appraisal of the final circuit behavior. While there are a large number of emulation-based fault injection methods in the literature, only those that mimic radiation effects through SRAM-based FPGAs are addressed in this thesis. So, they have been previously classified in [94] as Hardware-based and software-based fault injection methodology.

4.2.1 Hardware-based Fault Emulation

This technique is based on the use of an external hardware like a Joint Test Action Group (JTAG) controller, or through dynamic reconfiguration. Also known as reconfiguration-based approaches, they consist in a complete or partial modification of the configuration memory of the FPGA in order to inject faults in the CUT. Indeed, hardware-based emulations use reconfiguration process instead of adding extra logic in the CUT. Thus, they incur in no area overhead at the expense of a run-time overhead issued from the said reconfiguration process. For example, the tool in [5] proposes a SEU emulation platform that targets the configuration memory of an FPGA under test via partial reconfiguration. The input stimuli and output vector from a *golden system* are imported by the test-bench and then simulated in VHDL/Verilog to be finally compared with the FPGA's results. In [76], a fault-injection tool based on Tool Command Language (TCL) scripts accesses different resources of Altera's FPGA via a JTAG interface. In [36], a different types of faults such as stuck-at, bit-flip, pulse, indetermination, stuck-open, delay, short, open-line, and bridging are addressed. All these faults are injected by partial reconfiguration capabilities provided in Xilinx FPGAs. In [55], faults are injected using a technique known as read-modify-write applied to the configuration bits via partial reconfiguration. Recently, the authors in [114] proposed to use N-modular redundancy (NMR) for masking the effects of SEUs on FPGAs. Additionally, a soft-core PicoBlaze processor and a Xilinx ICAP interface are used to control the fault injection campaign, which is also performed through partial reconfiguration techniques.

4.2.2 Software-based Fault Emulation

Instead of using complex reconfiguration features which are not always available, other methods have proposed to insert faults through modification of the RTL model of the circuit. In general, they add a small fault injector circuit known as *saboteur* in sensitive parts of the CUT. The key idea behind saboteurs is to avoid time penalty [44, 48, 70, 73, 74]. Therefore, in software-based techniques, faults are directly injected to the RTL design, which significantly simplifies the injection process, allows the designer to specify which part of the CUT to test, and keeps the HDL of the CUT unchanged. For example, the work in [109] addresses permanent and transient fault injection in the flip-flops and the logic gates of the CUT. Fault injection is done by applying extra logic gates and wires to the original design description. The authors of [80]

implement saboteurs to inject SEUs, MBUs and stuck-at faults at the RTL level. Two versions of the CUT are implemented inside the FPGA to compare the results. In [23], a mask-chain and a combinational circuitry are added in the *netlist* description to inject faults in the flip-flops (a netlist is a list of interconnected blocks implementing the circuit logic which can be expressed in Verilog or VHDL). Similar software-based methodologies can be found in [10, 63, 67, 70]. Figure 4.1 illustrates and summarizes the classification provided in this section.

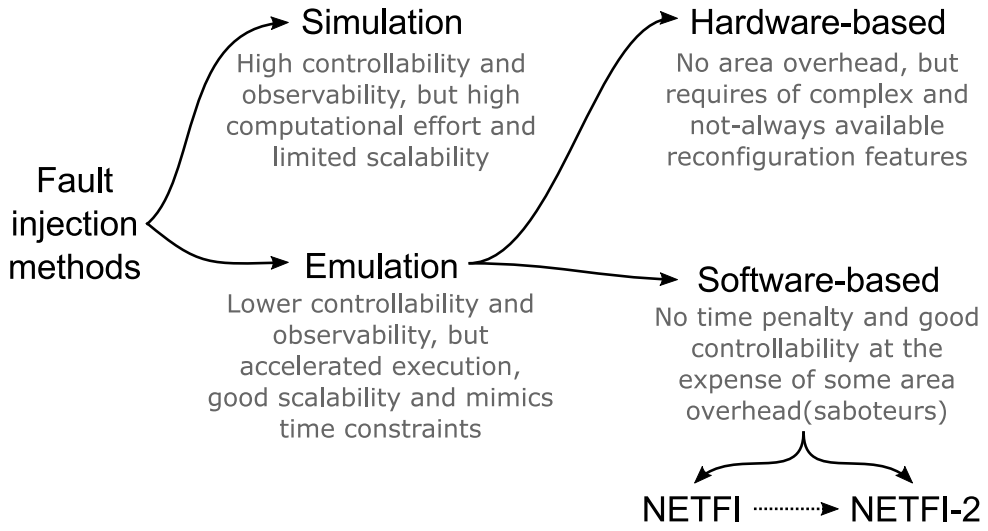

Fig. 4.1: Fault injection methods classification

Although there are a large number of methodologies in the literature that study the injection of SEUs or SETs at RTL level, only a few of them consider the effects of SEUs and SETs at the same time. One of them is NETFI [73, 74]. In NETFI, a modified netlist is integrated into a hardware-based platform such as THESIC+ or similar [48] to control the injection process during the experiment execution. However, the original NETFI procedure has been criticized for the large hardware overhead and the complexity of the required experiment controllers [105]. Also, the resulting accuracy is limited as SET injection depends on how logic gates are grouped in arbitrarily large FPGA combinatorial blocks [73]. NETFI-2, designed to overcome these two limitations, is described in the following section.

4.3 NETFI-2

NETFI-2 aims at taking the best of existing emulation solutions (including NETFI) to gather them all in a single and efficient methodology for both SEUs and SETs. In general, it can be cataloged as a software-based emulator that runs on a single FPGA [110]. As with the original NETFI, the injections of SEUs and SETs in NETFI-2 can be accomplished by non-invasive signals in the RTL design. This means that the functionality of the device (typically described in HDL) is not modified; thus resulting transparent for the user. In particular, NETFI-2 simplifies the campaign execution in terms of required hardware while facilitating the design of complex

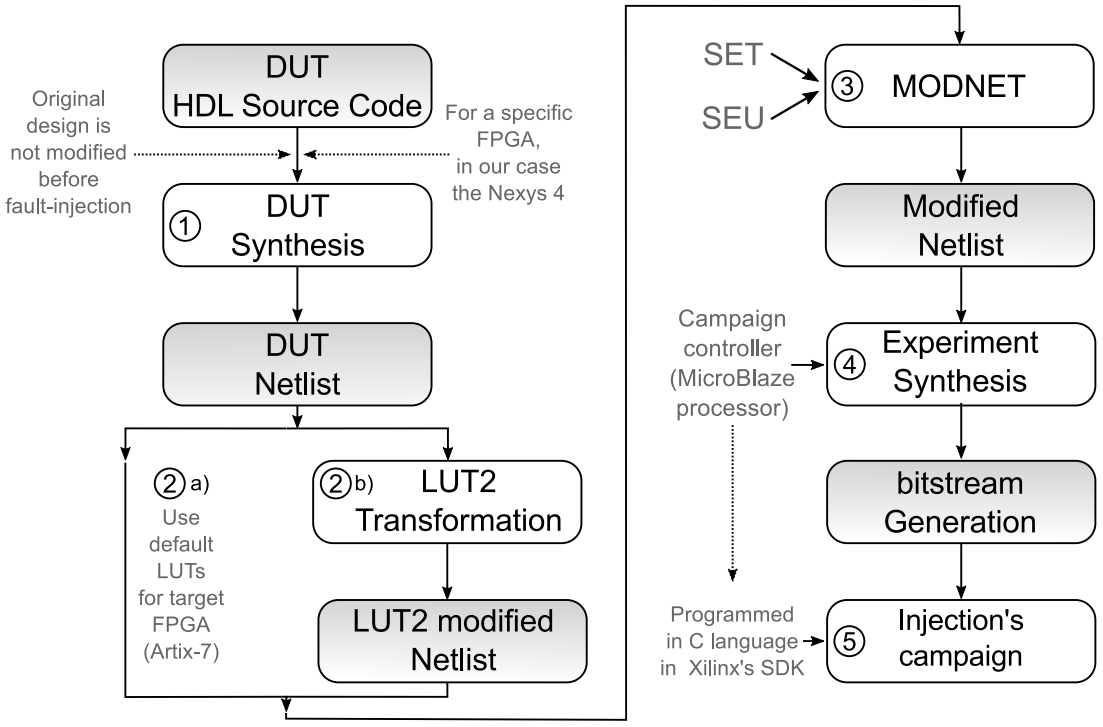

Fig. 4.2: NETFI-2 methodology

fault-injection campaigns. The latter is achieved by implementing the campaign controller in a soft-core processor sitting next to the CUT in the target FPGA. Furthermore, NETFI-2 allows to fine-tune the granularity of the combinatorial part of the design under test (which is provided in HDL), in order to more accurately mimic the underlying logic of the circuit and therefore, yield more accurate results when it comes to injecting SETs. More details about this are provided in the subsections below.

Also, by using NETFI-2, it is possible to determine which specific component is responsible for each particular error observed in the output. This unique feature allows to perform improvements in the circuit design to increase its robustness.

4.3.1 Methodology

Figure 4.2 illustrates the work-flow proposed for NETFI-2. Although similar to the one proposed for NETFI, several modifications discussed in this section make NETFI-2 a more flexible and accurate methodology.

Initially, the HDL description of the CUT is used to obtain a first synthesis in Step ① . It should be noted that, unlike other fault-injection methods, this first step does not require any modification in the original design. In this step, the synthesis of the CUT is targeted for a specific FPGA (an Xilinx Artix-7 in our proposed architecture, later detailed in subsection 4.3.2). Although different tools can be used to accomplish this step, throughout this Chapter we consider Synplify Premier from Synopsys as a software for FPGA synthesis because it allows to

export a netlist, which is necessary in further steps.

In step ② , a decision must be made regarding the underlying combinational blocks utilized to implement the CUT in the FPGA where the experiment will take place. Indeed, the size of the FPGA LUTs used for implementing the combinational logic of the CUT has a direct impact on the effect of the SETs on it (unlike SEUs, which are provoked in flip-flops components). Although LUTs with different sizes can be used, this work studies two possible cases: ② a) implementations based on a mixture of default LUTs sizes chosen by the synthesis tool (for the Artix-7, Synplify uses LUTs with 4, 5 and 6 inputs, also known as LUT4, LUT5 and LUT6 respectively); and ② b) implementations based only on LUTs with 2 inputs (LUT2). This step can either be performed by manual scripting over the obtained netlist or integrated with modifier tool, further described below. As discussed in subsection 4.3.3, limiting the synthesis tool to use only LUT2s result in a more granular approach for the SET fault-injection campaign.

In Step ③ , the netlist (either comprised by LUT2s or a combination of other types of LUTs) is used as input for the MODify NETlist (MODNET) tool. The output of MODNET is a modified (but functionally equivalent) netlist with a large number of extra input signals used to access all memory cells and logic blocks of the design to inject faults in the CUT [73]. The resulting synthesis of the modified netlist includes some additional combinational circuitry to the design, but the sequential circuitry is left unchanged. Details on both SEU and SET emulation are detailed below.

- **SEU Emulation:** The emulation of SEUs in a digital circuit requires to add some instrumentation hardware around the flip-flops of the design in order to perturb their content at any given moment (decided by the campaign controller). To this end, a functionally equivalent structural design of the CUT is obtained (i.e., in terms of Xilinx primitives) and the hardware inserted around the flip-flops of said CUT will depend on if the flip-flop features enable signal or not (Figure 4.3). Thus, flip-flops with enable signal are always left in sleep mode unless its enable signal is activated. In this case, an injection (*inj*) signal combined with some additional logic (a XOR-gate and a multiplexer) are used to inject faults. Flip-flops without enable signal are always in active mode. In this case, an *inj* signal is used to select which input, with or without fault, should be registered by the flip-flop in the next clock cycle.

- **SET Emulation:** In the case of SETs, MODNET modifies the LUTs and all the logic gates of the netlist by simply adding an extra multiplexer at the output to select the appropriate value (erroneous or correct). However, larger LUTs would be treated as a single point of failure, even if in the final implementation of the device is based on simpler modules. ASICs will probably be manufactured following an approach based on an Uncommitted Logic Array (ULA), where a set of prefabricated NAND-gates are later interconnected in a customized manner by adding metal layers in the CMOS design [96].

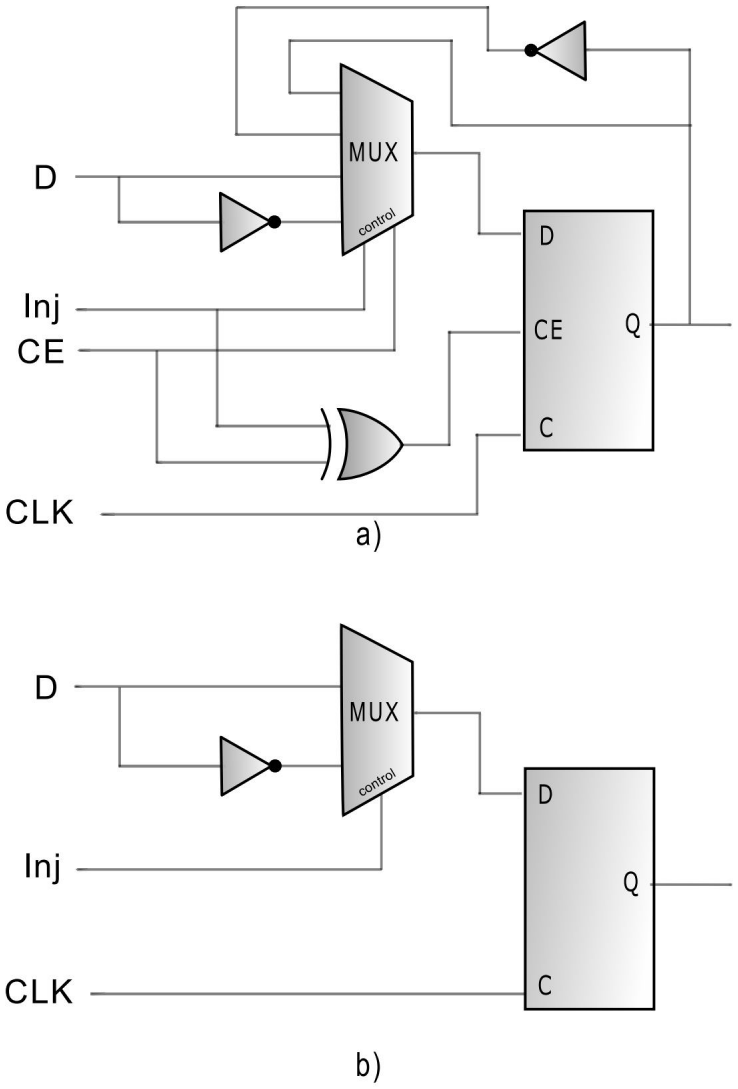

Fig. 4.3: Modification of flip-flops with enable signal in a) and without enable signal in b) [73]

Semi-custom or full-custom designs based on NAND-gates are also possible if the circuit will be fabricated in extremely high volumes [79]. In any case, a 2-input NAND-gate is functionally equivalent to a LUT2. To better mimic the final CUT behavior, Step ② allows to specify proper LUT sizes. Specific details and LUT transformation strategies are given in Section 4.3.3. Whichever the case, the resulting modified netlist can be seen as a different version of the original one, including signals to access the sensitive elements (flip-flops and LUTs), either to obtain their value or to inject faults.

In Step ④ , a campaign controller is integrated within the modified netlist for the target FPGA. In NETFI-2, the campaign controller is implemented in a soft-core processor that is in charge of managing the SEU and SET fault injection campaign by being directly wired to the CUT modified by MODNET. To this end, the netlist obtained in Step ③ can be synthesized in the Electronic Design Interchange Format (EDIF) and then attached to the processor via a direct interface. The resulting *bitstream* implementing the complete circuit (controller and

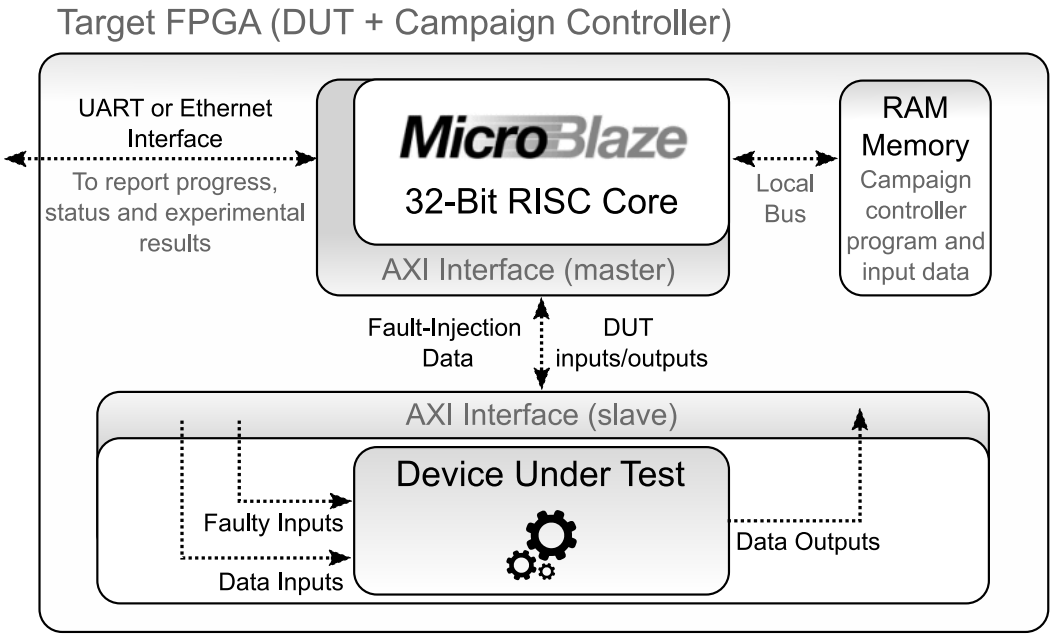

Fig. 4.4: NETFI-2 architecture

CUT) is thus generated and implemented in the target FPGA.

Finally, the experiment in Step ⑤ can be directly executed from the soft-core processor without requiring additional or external hardware support. Indeed, the whole SEU and SET fault-injection campaign (including the post-processing of the results), can be conveniently encoded in the processor software. By accessing high-speed interfaces connecting the CUT, the software can efficiently execute several iterations of fault-injection experiments with different data inputs and fault points. The latter process is flexible enough to also include multiple simultaneous injections to execute extensive MBUs fault-injection campaigns. Given that injections and outputs in MBUs campaigns can be very large, executing the configuration, filtering and result post-processing within the embedded controller processor becomes an important advantage of NETFI-2.

4.3.2 Architecture

As previously stated, NETFI-2 is integrated into a single FPGA where the CUT and the experiment controller are instantiated and connected by a dedicated interface. To this end, the Advance eXtensible Interface (AXI) has been used, since it has been adopted by Xilinx for implementation of complex System-On-Chip (SoC) designs. In this particular case, the MicroBlaze processor (campaign controller) is a master, whereas the CUT is a slave. Figure 4.4 shows the presented architecture implementing the NETFI-2 methodology.

The Xilinx MicroBlaze processor is based on a 32-bit RISC architecture allowing the cam-

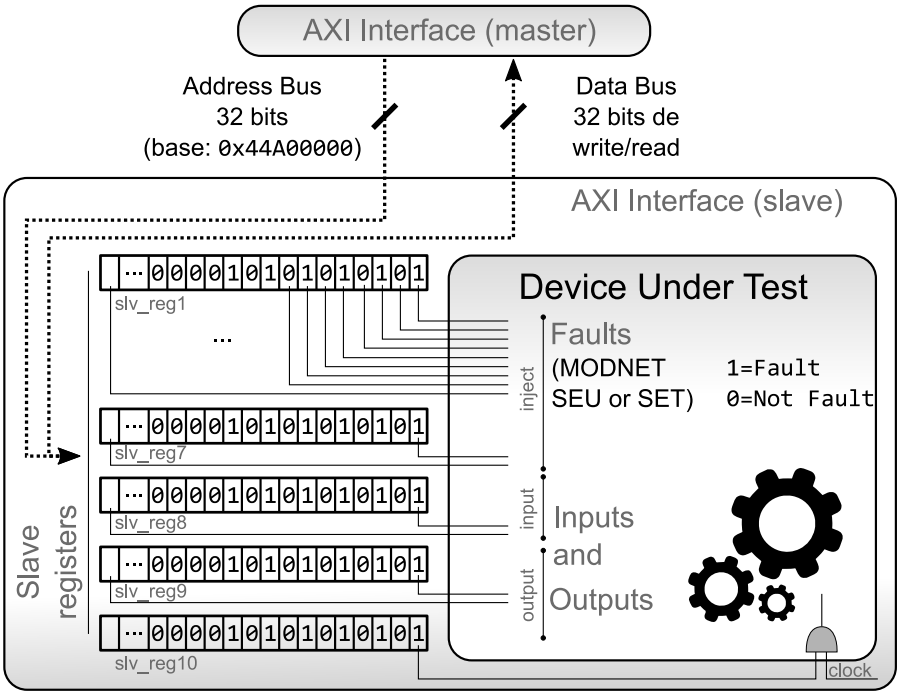

Fig. 4.5: Interface with CUT

paign controller to have an approximate rate of one instruction per clock cycle [64]. This is an important feature for keeping the processor software and CUT synchronized (they can, ideally, share the same clock). The architecture of the processor can be easily customized and extended using the tools provided by Xilinx. For example, to access the system memory, the processor uses dedicated buses for instructions and data which frees the other bus loads. This is of particular importance for NETFI-2, since it allows to have an interface based on the AXI protocol fully devoted to interact with the execution of the fault-injection experiment.

The MicroBlaze processor can make use of an UART or Ethernet interface to communicate with the outside to report the status of the experiment or its final results. Unlike previous work with external controllers [105], the process of fault injection in NETFI-2 does not necessarily depend on this communication protocol. In contrast, the fault injection campaign can run autonomously within the target FPGA while reporting partial or total results only when the controller has available processing capacity. Therefore, the execution speed of the experiment is not compromised by the external communication protocol.

The AXI interface allows to communicate a slave CUT with a master unit (campaign controller) that exchange information while using minimal area in the FPGA. The master unit uses memory mapping to read or write values into 32-bit registers contained in the slave device as illustrated in Figure 4.5. According to the specification offered by Xilinx, up to 256 transmissions can be performed in bursts [123], improving the campaign execution speed. NETFI-2 uses this communication channel to configure the SEU and SET fault injections in the components already intervened by MODNET (*inject* signals), as well as to configure the input and read the

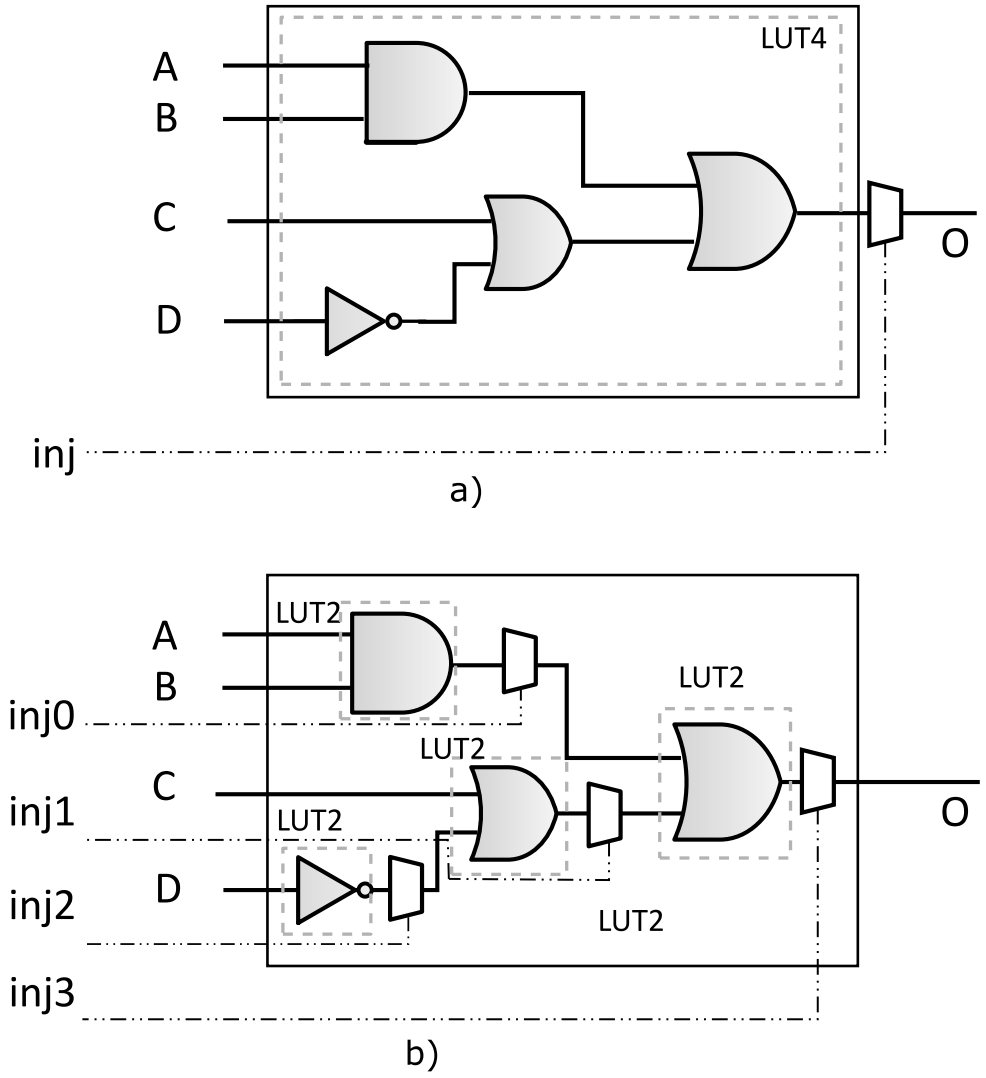

Fig. 4.6: Example circuit a) based on a LUT4 and b) based on LUT2s

output values of the CUT. Since the AXI slave device maps the memory addresses of the microprocessor to the CUT fault-injection signals, the campaign can be conveniently defined by software. Furthermore, the clock of the CUT can also be managed via this interface (either if the CUT's clock is enabled or disabled). It is worth noticing that a clock divider might be also included in this part of the architecture to accommodate different CUT operating frequencies.

4.3.3 LUT Transformation

To model SETs, an extra multiplexer at the output of each LUTs is added to select the appropriate output (error or not error). However, as previously mentioned, one should pay attention to the fact that the synthesizer may group many logical gates in one LUT, implying that the injection will only target the output of the output gate instead of each of them individually. It should be noted that this is not necessarily a disadvantage if the final implementation of the CUT is an FPGA with similar LUT sizes than the target FPGA. Although the original method

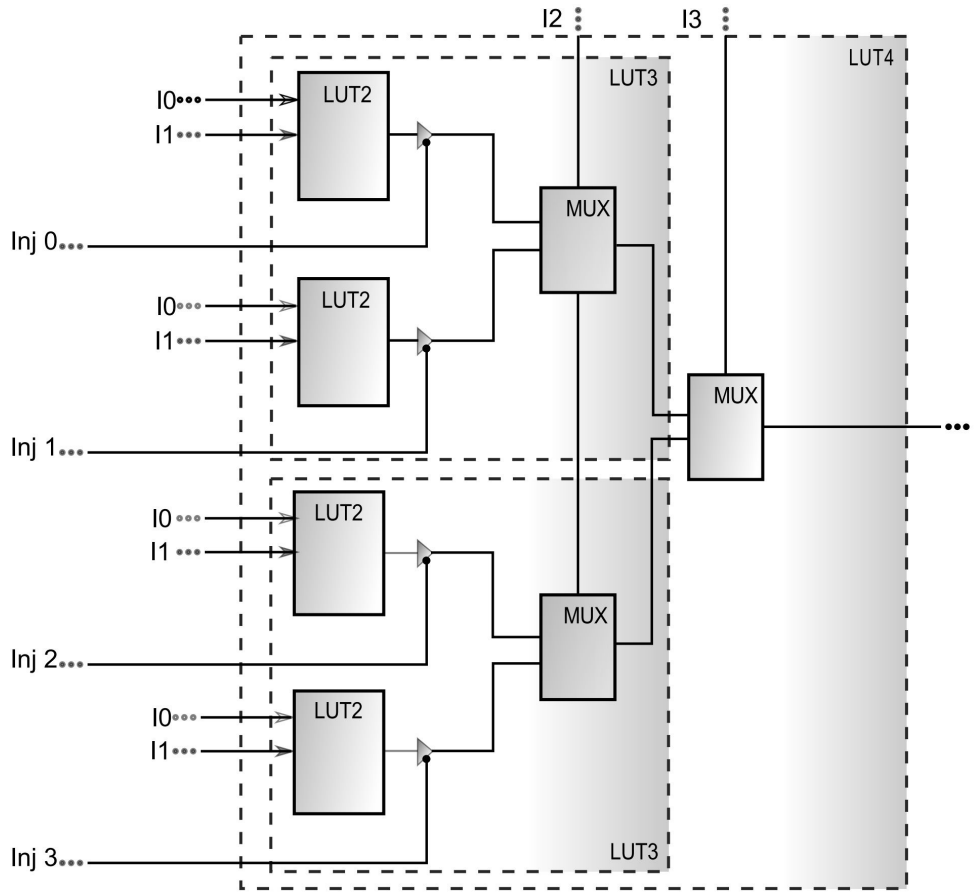

Fig. 4.7: Implementing a LUT4 with four LUT2s and three multiplexers or two LUT3s and two multiplexers

with the default LUT configuration is assumed to give a good feedback concerning any fault tolerant design [73], its accuracy can be compromised when considering implementations in ASICs (typically based on simpler gates [96]).

In order to enhance the granularity and accuracy of the SET fault-injection methodology, NETFI-2 proposes an alternative step ② b) as illustrated in Figure 4.2. In this step, the LUTs of the functionally equivalent structural design interfaced for SEU emulation (see Subsection and Figure 4.3) are also intervened in order to create a CUT that is only implemented using LUTs of specific sizes. In our case, two-input LUTs (LUT2) are considered as they can unequivocally mapped to 2-input gates, such as the ones typically used in ASIC implementations. Therefore, when using this netlist in MODNET, SET fault-injection signals will be added for elemental circuits components at the gate level. On the other hand, SEU injection signals will still be attached to flip-flops, which are independent of combinational elements implemented in LUTs.

The final goal of transforming the LUTs is to increase the granularity of the SET fault injection. The example in Figure 4.6 a) illustrates that a given circuit can be implemented with a single LUT4, while Figure 4.6 b) shows that it can also be implemented using LUT2s. When modified by MODNET, the former implementation will allow for a single SET injection signal

at the output of the LUT4 (*inj* signal), while the latter will let four different faults to be injected in the CUT (*inj0* up to *inj4* signals). In general, if considering that every injection is set to zero, the combinational logic provides the same output for both implementations. However, if a fault-injection campaign is executed on each of these different circuits, different error rate estimations would be obtained. For example, if faults are injected in the LUT4-based circuit, only one fault can be injected, whereas in the LUT2-based circuit, 4 different sensible points can be evaluated. An injection in the first sensible point of the LUT2-based solution (*inj0*) will have no effect in the output when either C is set to one or D is set to zero. Indeed, the output O goes high for any of the inputs of the final OR-gate being set to one. Similar effects are observed for *inj1* signal if A and B are set to 1 since the value at the output of the circuit (O) will always be one. On the other hand, the injection in the final output (*inj4*) would have a direct impact in the output as it always changes the correct result of the circuit. In this case, whichever the A, B, C, D input values are, an error would be observed at the output when signal *inj4* is set. In general, a LUT2-based circuit with a single output like the example will, at least, evidence a 25% SET error rate in a fault-injection campaign. However, the functionally equivalent implementation of Figure 6.a) will always deliver an error in the output (i.e., 100% SET error rate). Therefore, a suitable LUT transformation must be chosen to have an accurate SET error rate measurement of the CUT.

In general, whichever the size of the LUTs chosen by the synthesis process, there is a systematic way of transforming them to functionally equivalent implementations based on LUT2s. This can be achieved by combining LUT2s with multiplexers as shown in Figure 4.7, where a generic LUT4 (with inputs *I0*, *I1*, *I2*, *I3*) can be transformed to a LUT2-based implementation. Also, LUT3 construction blocks are illustrated for similar transformations to LUTs of three inputs.

A non-minor concern from a user perspective is to determine the optimal LUT transformation solution. This is not always a straightforward answer. For example, if the final environment for the CUT is the same FPGA where the experiment will be executed, then letting the synthesizer to decide the LUT size would render more realistic results. However, in case the design will later be ported to an ASIC implemented following the aforementioned ULA approach [96], or even a full-custom design [79], then a LUT2-based approach will provide more accurate error-rate estimations. Further analysis would still be required to properly understand how the LUT2 solution maps to other modules used by the chip's manufacturer.

4.3.4 Evaluation and Validation

In order to evaluate and validate NETFI-2, two CUTs are proposed as case study (i.e., a Stochastic Bayesian Machine (SBM) and a Support Vector Machine (BM)). Both **SBM** and **SVM** were modified by MODNET and then submitted to a SEU and SET fault injection campaigns using the described NETFI-2 architecture. The time required by the fault-injection campaign as well

as the impact of LUT transformation in the SET results are analyzed. Also, the results from a radiation test experiment performed in the SVM was included in this section. The idea here was to compare the results between our emulation fault injection tools with the real radiation test results.

4.4 Baysian Machine Under Test

In this work, a module of a Stochastic Bayesian Machine (SBM) implemented in VHDL is proposed as CUT. A SBM is a type of stochastic compiler capable of calculating Bayesian inferences based on a set of probability distributions presented at their input [47]. These machines base their internal architecture on variables represented as stochastic *bitstreams* (notice that this is not an FPGA bitstream) which can make them intrinsically resistant to failures [4]. In particular, the proposed SBM is composed of small parallel modules called *BM-slices*. The BM-slice can be implemented in hardware (for instance, an FPGA [38]) using the circuit shown in Figure 4.8. A BM-slice computes the stochastic signal encoding $P'(M = i)$, for a given i, out of its 13 input bitstreams. As a result, at any clock signal, the size of the input of a BM-slice is 13 bits (hence, 8192 possible input values) while the size of the output is 1 bit. Besides, the BM-slice also includes a clock and reset port. The BM-slice circuit is implemented using AND-gates to perform stochastic multiplications and stochastic adders to calculate the addition of probabilities encoded in the bitstream signals. The stochastic adders are in turn implemented by comparators, multiplexers and counters making the BM-slice an interesting sequential and combinatorial circuit to study under SEU and SET effects. Nonetheless, it is worth noticing that full SBMs instance n BM-slices with this architecture working in parallel to operate over bitstreams of $n \times 13$ elements. The hypothesis held in [47] is that the complete system composed of n BM-slices should be intrinsically robust against SETs and SEUs. The interested reader can refer to [26] for a complete robustness evaluation of a full SBM. However, in this work the robustness of an individual BM-slice (and not a full SBM) will be evaluated.

4.4.1 BM-slice LUT Transformation

The BM-slice netlist is processed by MODNET to obtain a netlist with components sensitive to SETs and SEUs, and interfaced with injection signals. The MODNET tool identified 82 sensible points for SET injection when synthesizing this CUT in the default LUT configuration chosen by Synplify (of size 4, 5 and 6 as result of synthesis for the Artix-7). However, these 82 sensible points become 812 sensible points when implementing the circuit using LUT2-only transformation. Indeed, this represents a SET fault-injection granularity almost 10 times greater for circuits implemented in smaller 2-input gates. The impact on the error rate resulting from this transformation is later studied in Section 4.4.3. Nevertheless, the gain in accuracy comes

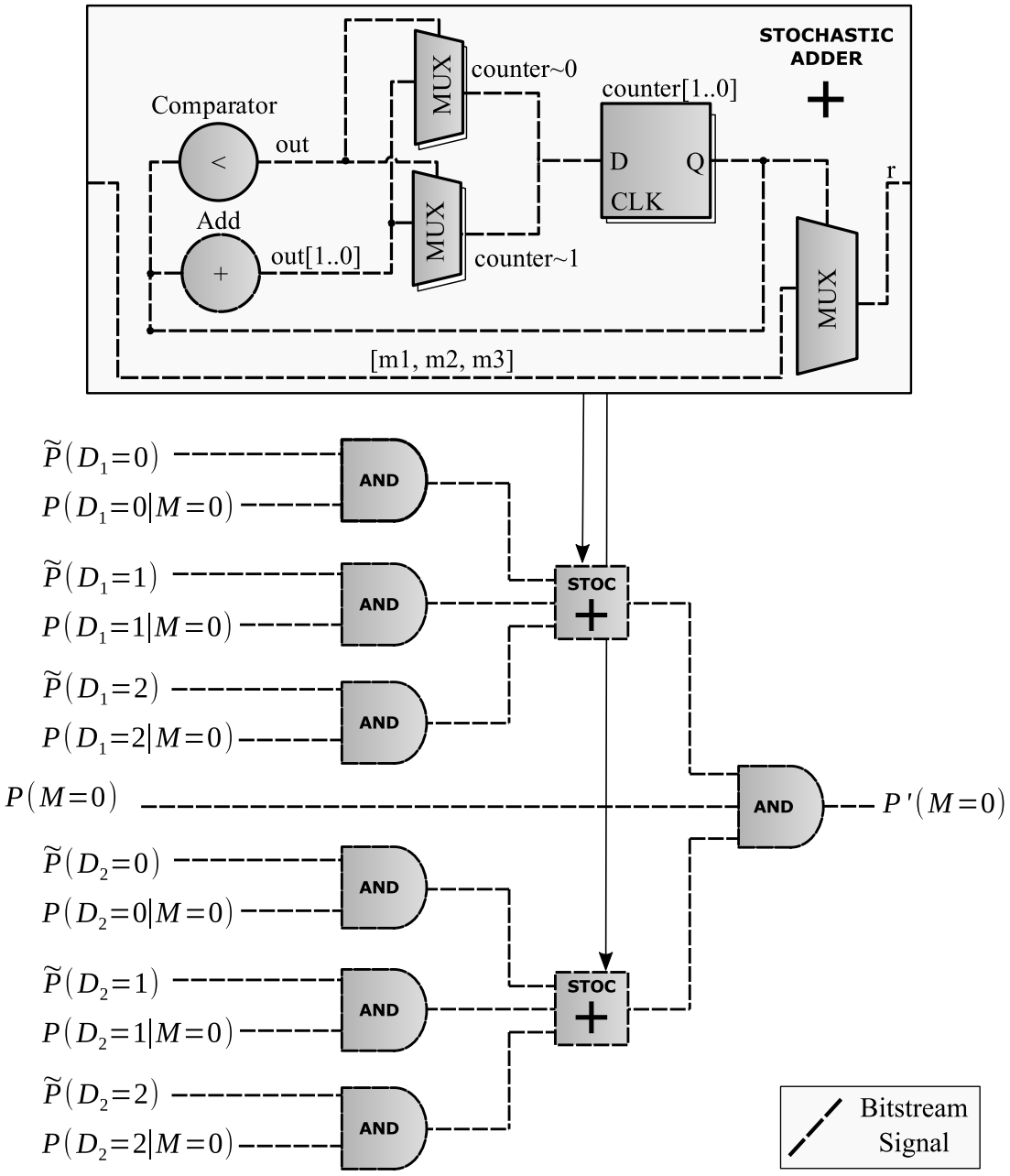

Fig. 4.8: BM Slice - Hardware implementation

at an extra cost in terms of FPGA resources. Table 4.1 summarizes the resource utilization in the Artix-7 FPGA board in terms of slice LUTs utilization including the MicroBlaze processor. The presented data of resources utilization were taken out of Vivado's reporting utility, post implementation. The Artix-7 slice LUTs increment from 1961 to 2502 (+27.9%) when implementing a SET fault-injection campaign based on the LUT2 transformation, compared to the SET fault-injection campaign based on the default LUT sizes (sizes 4, 5 and 6). Utilized registers also increase (+36.9%) as the controller to the CUT interface must accommodate several extra injection signals. Since FPGA slices contain groups of several types of LUTs, flip-flops and multiplexers, there is not always a linear correlation between the LUT increment

form LUT4-5-6 to LUT2 transformation and the one measured in real FPGA synthesis reports. Indeed, the final resource utilization in a given FPGA platform will strictly depend on its inner architecture combined with the optimization strategies taken by each manufacturer synthesis tool.

Table 4.1: Detail of resource utilization in the FPGA

	# Used	Available	% Used
	MicroBlaze		
Slice LUTs	1746	63400	2.75%
Slice Registers	1475	126800	1.16%
	MicroBlaze + BM (LUT4-5-6)		
Slice LUTs	1953	63400	3.08%
Slice Registers	1866	126800	1.47%
	MicroBlaze + BM (LUT4-5-6) + Injection SET		
Slice LUTs	1961	63400	3.09%
Slice Registers	1868	126800	1.47%
	MicroBlaze + BM (LUT4-5-6) + Injection SEU		
Slice LUTs	2074	63400	3.27%
Slice Registers	1839	126800	1.45%
	MicroBlaze + BM (LUT2)		
Slice LUTs	2140	63400	3.38%
Slice Registers	2493	126800	1.97%
	MicroBlaze + BM (LUT2) + Injection SET		
Slice LUTs	2502	63400	3.95%
Slice Registers	2558	126800	2.02%

4.4.2 Fault-Injection Campaign

As previously stated, a fault injection campaign was carried out with NETFI-2, on a Nexys 4 board which is equipped with a Xilinx Artix-7 XC7A100T-CS324 FPGA. The synthesis of the modified CUT was accomplished with Synopsys Synplify-Premier tool to obtain the EDIF file. The processor synthesis and bitstream generation of the FPGA were made using the Vivado tool. To run this campaign, the original MODNET tool was updated accordingly in order to support state-of-the-art Xilinx FPGAs. Implementation results summarized in Table 4.1 show that the MicroBlaze processor used as the controller of the experiment, only requires 2.75% of the slice LUTs and 1.16% of slice registers of those available in the target FPGA. In other

words, the controller overhead in NETFI-2 is low enough to allow the majority of the FPGA resources to be used for large CUT implementations.

Table 4.2: Extra signals presented in BM-slice

	SEU signals	SET signals
LUT4-5-6	27	82
LUT2	27	812

For the fault injection campaign, a specific application had to be developed to run under the MicroBlaze soft processor. The designed algorithm for this case study is illustrated in Algorithm 2. Xilinx provides a dedicated Software Development Kit (SDK) to conveniently accomplish the implementation of the program. The tool also provides drivers and libraries to elaborate the embedded code (in C language). As shown in Table 4.2, besides the 13 bits of BM-slice inputs (8192 possible inputs values), the CUT also includes 27 signals for injecting SEUs and 82 signals for injecting SETs when implemented in LUT4-5-6 (or 812 when using LUT2-only injections - Algorithm 2 depicts this case -). In order to extensively study the device's robustness, it is necessary to perform a fault injection in every possible location (i.e., every bit of the injection's port), for every given input. Therefore, the injection's campaign program loops for every input, making an injection (setting a bit to 1 in the corresponding slave register) in all the available locations of the 812 bits (SET injections for LUT2-implementation), 82 bits (SET injections for default LUT implementation) or 27 bits (SEU injections) of the injection ports. Since the width of the slave registers provided by the AXI Interface is 32 bits, it is necessary to use several of them to cover all the possible injection points (for instance, 26 registers are needed for the 812 SET injection points in the LUT2 implementation). The last register in the LUT2 CUT implementation, only has 12 addressable bits to perform an injection; thus, this register is left out of the initial loop (that goes through the entire 32 bits) and addressed separately in Algorithm 2, in Lines 15-22. It should be noticed that setting several bits in the registers would imply injecting MBUs, which is a new feature of NETFI-2 with respect to its predecessor [72]. In order to determine when an error occurs, all the injections are initially set to zero (i.e., no fault injections performed in the device) to then sample the golden output and store it in a temporal variable ($output_1$). Then, for each bit in all the slave registers, an injection is performed by changing the bit (Lines 7 and 16), the output of the circuit is retrieved (Lines 8 and 17), the injection vector is restored (Lines 12 and 21), and finally, in case of mismatch with the golden result, the error count is updated (Lines 9-11 and 18-20). Before switching to the next input value (variable i), $error_i$ is summed to the total error count present on the device ($error_t$), and reseted to zero.

Algorithme 2 Pseudo-code for SET injection campaigns of LUT2-based implementations

```
1: for i = 0 to 8191 do                                   ▷ Total amount of inputs
2:     injections = 0
3:     input = i
4:     output_1 ← CUToutput
5:     for slvreg = 0 to 26 do                            ▷ slave registers
6:         for j = 0 to 31 do                             ▷ 32 bit size register
7:             slvreg[j] ← slvreg[j] XOR 1
8:             output_2 ← CUToutput
9:             if output_1 ≠ output_2 then
10:                error_i + +
11:            end if
12:            slvreg[j] ← slvreg[j] XOR 1
13:        end for
14:    end for
15:    for k = 0 to 11 do                                 ▷ last inj register size
16:        slvreg[k] ← slvreg[k] XOR 1
17:        output ← CUToutput
18:        if output_1 ≠ output_2 then
19:            error_i + +
20:        end if
21:        slvreg[k] ← slvreg[k] XOR 1
22:    end for
23:    error_t = error_t + error_i                        ▷ Accumulate errors
24:    error_i = 0                                        ▷ Reset error_i value
25: end for
```

4.4.3 Result Analysis

The error rates for SEU and SET results obtained from the fault injection campaign, using LUT2s and LUT4-5-6, are summarized in Table 4.3. It can be observed that the fault injection campaign based on LUT2 transformation shows a decrement of the error rate as discussed in Section 4.3.3. In particular, the increase of injection points will eventually derive in faults that finally do not incur in any error thus lowering the error rate metric. In this table, time measurements made for fault injection in the BM-slice circuit can be seen, which in both cases can be practically disregarded. In preliminary evaluations of NETFI-2, simulation-based fault injections campaigns were also conducted with comparison purposes [110]. These fault injection campaigns took more than 300 seconds to run on the same BM-slice design, highlighting the speed-up of emulation-based solutions such as NETFI-2.

When taking a closer look at the results of the SET fault-injection campaign, it is possible to identify the LUTs responsible for the greatest amount of errors at the output. In order to facilitate the observation, Figure 4.9 shows the LUTs of sizes 4, 5 and 6 with higher impact in the SET final error rate. Analyzing the location of the outlier LUT (LUT 6 with ID 81), it

Table 4.3: Fault Injection Campaign Results

	Error rate SEU	Error rate SET	Execution time SEU/SET
NETFI-2 LUT4,5,6	11.30%	30.10%	<1 s
NETFI-2 LUT2	11.30%	11.50%	<1 s

was found that it was the building block in the final stage of the circuit implementing the CUT. This means that every output of the other LUTs in the CUT data-path passes by this final LUT, making it a particular weak point responsible for a significant number of errors. Also, it can be observed that the rest of the LUTs shown in Figure 4.9 are in the range of ID 40 to ID 61, which, in our particular CUT implementation, are those located within the final stages of the circuit. With this in mind, and as general statement, it can be assumed that LUTs located at final stages have a higher impact in the final error rate of the CUT. The added value of NETFI-2 in this context is that the methodology allows to precisely measure and anticipate this phenomena.

On the other hand, if the CUT is implemented by applying the LUT2 transformation, the IDs of the final 16 LUT2 (those that implement the same combinational part implemented by LUT6 ID 81) range from ID 796 to 811 (out of a total of 812 injection points). By using these results, it was observed that out of these 16 LUT2, those with ID 796, 800, 804 and 809 are responsible for the 33.3% of the total amount of errors produced by LUT6 ID 81 in the LUT4-5-6 implementation. This means that the LUT2 transformation approach can be used to accurately identify weak combinational points in the design, allowing to perform detailed improvements, changes or hardening implementations on the circuit layout. Indeed, the latter would incur in more efficient and granular hardware configurations when studied in LUT2-only implementations (it might be more convenient to harden a few LUT2s than a larger LUT6). A Popular hardening technique to achieve the latter is known as Triple Modular Redundancy (TMR), which can be properly defined and addressed by using NETFI-2. Although not described in this section, similar analysis can be performed based on SEU injections by identifying most critical flip-flops in the CUT.

4.4.4 Discussion

We have analyzed the fault injection campaign performed in the Bayesian Machine in three different points of view, which are listed below:

- **Efficiency and Flexibility:** One of the main advantages of NETFI-2 is that is capable of being implemented in FPGAs that are not particularly expensive, like the Artix-7. This is not a minor feature given that similar methods sometimes require high-end FPGAs (sometimes many of them) to accomplish the same results. Indeed, NETFI-2 flexibil-

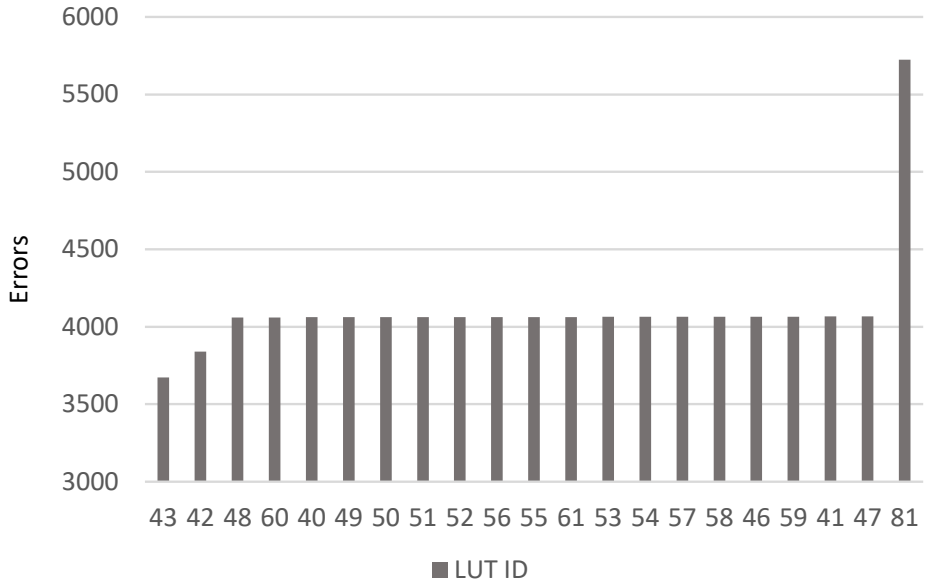

Fig. 4.9: LUTs of BM-slice (of sizes 4, 5 and 6) where the highest amount of errors were observed

ity and efficiency come from the fact that the complete fault-injection campaign can be translated to a software code executed by a soft-core implemented side-by-side with the CUT. Another consequence of the latter is that CUTs can be conveniently controlled and configured under different LUTs and flip-flop technologies.

- **Implementation Exploration:** As previously discussed, a LUT4 will always deliver one error per fault (100% error rate). On the other hand, a LUT2 equivalent transformation of a LUT4 can render from 0 to 4 errors per each four faults injected (0%, 25%, 50%, 75% or 100% error rate). If the enumerate is between 0% and 75%, then one, two or three LUT2s respectively can be hardened by using individual TMR. However, if it is 100%, then it is better to implement a LUT4 and directly harden it. Indeed, if the design flow of the circuit under test allows for it, the designer can use NETFI-2 to explore different implementation possibilities combined with different hardening strategies. This is a unique feature of the NETFI-2 methodology in the context of radiation-hardened designs. It is worth noticing that, although it is accurate for some implementations, the LUT2 transformation demands an extra effort that can not be left unaccounted when working with larger CUTs. This effect was not seen in the BM-slice case study based on a simple test circuit. This effort not only involves FGPA resources (noticeably multiplexers) but also computation time in performing, analyzing and post-processing all the injections.

- **Implementation Analysis:** In general, the circuit designer has a predefined technology on which the device will be implemented (either a specific FPGA or an ASIC with a well-defined type of combinational modules or logic gates). In these cases, in order to

determine which LUT transformation is better suited to analyze and configure the fault-injection, it is necessary to study how the final deployment of the CUT will be made. In the case of an ASIC implementation based in two-input logic gates, the LUT2 transformation brings to the table an unprecedented level of accuracy. ASICs manufactured with multiple-input logic gates should be further studied by the designer to determine the most suitable LUT transformation technique in NETFI-2. Indeed, there is not a general answer, nor a one-fit-all solution for ASIC-based designs. On the contrary, if the final deployment is an FPGA similar to the one used in the fault-injection campaign, the default LUTs chosen at synthesis time will deliver the most accurate results. In case there is not enough information on the final implementation of the circuit, then NETFI-2 can only provide a general yet informative analysis on the reliability towards radiation effects. Evidently, the reliability of a circuit is strictly related to its final implementation characteristics.

4.5 Support Vector Machine Under Test

In this section, we have adopted the Support Vector Machine (SVM) as a case study for NETFI-2. Also, we have assessed the capacity of the SVM architecture to tolerate transient faults. And finally, we have provided a comparison between the emulation fault injection results with the radiation test results.

4.5.1 Support Vector Machine background

SVM is a classification algorithm belonging to the group of supervised machine learning techniques [116]. The algorithm addresses the problem of binary classification, i.e. a problem in which an observation (herein an input vector) has to be classified in one of two possible classes. Being a supervised machine learning technique, its workflow requires two phases, each one performed with a different algorithm: one for training (Sequential Minimal Optimization (SMO) algorithm) and another for classification (SVM algorithm).

Figure 4.10 presents an application of a SVM-based classification able to indicate whether an astronaut reaches a risky condition to have a cardiac problem – for instance, according to her/his heartbeat rate while moving or speeding on a treadmill in a space station.

At the training phase, the SMO algorithm uses the training vectors to calculate the weights of the linear function that better separates input vectors from the two classes, for example in Fig. 4.10: class “No heart problem" and class “Has heart problem", and the training vectors respectively represented by blue dots and red stars. The calculated weights, therefore, model a linear classifier dividing the elements of the two classes, i.e. they model the SVM algorithm equation.

Formally, a tuple $(\vec{x_i}, y_i)$ defines a training vector, in which the support vector $\vec{x_i} \in \mathbb{R}^2$ in

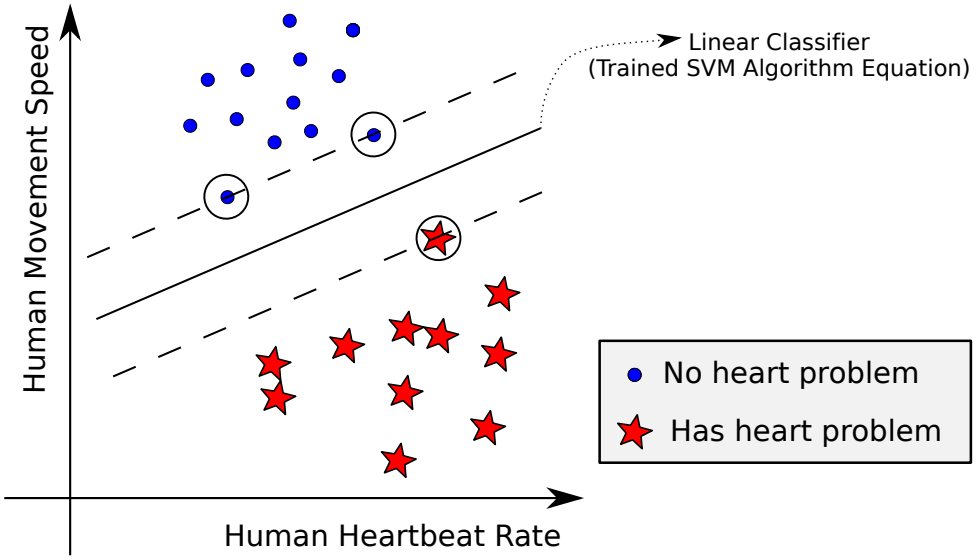

Fig. 4.10: An SVM algorithm equation (linear classifier) trained to classify the heartbeat condition. The horizontal axis represents the human heartbeat rate, while the vertical axis represents the human movement speed.

the example: heartbeat rate and speed features – both belonging to the set of real numbers – that are measurements performed on a person i. Moreover, the class $y_i \in \{-1, 1\}$, and in the example: -1 and 1 represent respectively the class “No heart problem" and the class “Has heart problem".

At the classification phase, the trained SVM algorithm is able to classify an input vector whose class is unknown, e.g. if the heartbeat rate and speed of an astronaut running on a treadmill indicate either “No heart problem" or “Has heart problem". The SVM algorithm equation to classify an input vector $\vec{x}$ with an unknown class is thus defined as:

$$score(\vec{x}) = \sum_{i=1}^{n} \alpha_i y_i (\vec{x_i} \cdot \vec{x}) + b \tag{4.1}$$

Each α_i is a weight, found at the training phase and associated with its respective training vector $(\vec{x_i}, y_i)$. The weights α_i shape the linear classifier (i.e. the SVM algorithm equation). At the training phase, the SMO algorithm also calculates the bias factor b. The sign of the $score(\vec{x})$ determines the class of the input vector $\vec{x}$, in the example: positive for “Has heart problem" and negative for “No heart problem".

Here, we focus on assessing only the classification phase of the SVM algorithm in hardware. As our dataset is linearly separable, further described in Subsection 4.5.2, we opted for the approach presented in [124] with a first order, i.e. linear, classifier to explore its benefits in terms of performance. Figure 4.11 shows the CUT of the SVM architecture. In addition, we have calculated the $\alpha_i y_i$ products beforehand, reducing one multiplication for each SV $\vec{x_i}$, and further optimizing the SVM architecture. The final implementation is completely combinatorial. The circuit is composed by three main parts: the Multipliers, the Adders and the Output, as

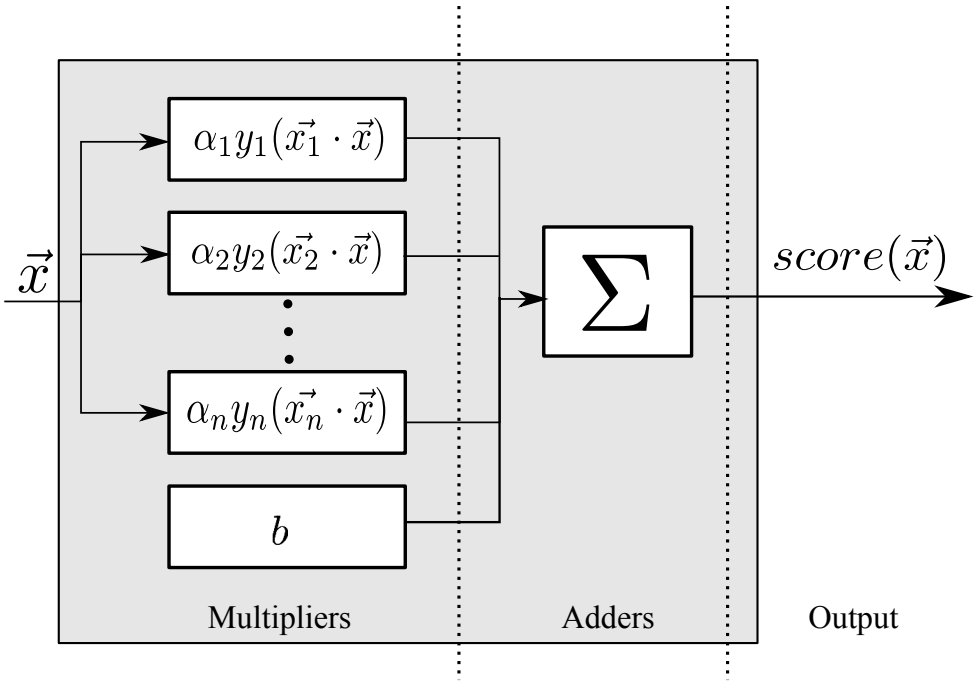

Fig. 4.11: Overview of the hardware-implemented SVM architecture design.

illustrated in Figure 4.11.

The SVM architecture adopted a 16-bit fixed-point representation, 8 bits being dedicated to representing the real part. Fixed-point representation was used in [99,120,124] as it is faster and provides less area overhead in comparison to floating point operations. Through simulations, we have confirmed that the representation is enough to avoid possible overflows and also to maintain sufficient precision. The primary inputs are composed of 32 bits (16 for each dimension of the input vector). The primary outputs are composed of 49 bits to maintain the calculation precision.

4.5.2 Set of input vectors

The target set of input vectors (dataset) was obtained from Monte-Carlo simulations representing current peaks and global delays obtained from golden integrated circuits (ICs) and faulty ICs [1]. The input vector is 2-dimensional, and 150 input vectors have been obtained from golden IC samples and 150 input vectors from faulty IC samples. The dimensions are thus:

- **Dimension 1**: global delay
- **Dimension 2**: current peak

This set of input vectors is sufficient to distinguish faulty asynchronous IC samples from fault-free asynchronous IC samples [1]. The set of input vectors has been partitioned into 2 subsets of the same size, one for training and another for classification, each one with 75 golden

Table 4.4: Resource utilization of the PL (Artix-7)

Resource	Utilization	Elements
Flip-Flops (FFs)	1.65 %	1751
Digital Signal Processing units (DSPs)	40%	88
Look-Up Tables (LUTs)	7.24%	3854

IC samples and 75 faulty IC samples. A SVM model has been generated by using MATLAB. From this model, we have obtained the α's and their respective support vectors $\vec{x_i}$. In total, 50 support vectors $\vec{x_i}$ have been generated at the training phase.

The CUT is a fully combinational SVM architecture design using fixed-point representation for its support vectors $\vec{x_i}$ and wights α_i (cf. subsection 4.5.1). The target platform is a Zynq-7000 [122], which is composed of two main parts: the Processing System (PS), consisting of an ARM Cortex-A9, and the Programmable Logic (PL), a Xilinx Artix-7 FPGA. The SVM architecture was implemented in PL part by using VHDL. The resource utilization of the PL is shown in Table 4.4.

The SVM algorithm depends fundamentally on multiplications. Most modern FPGAs, including the Artix-7, have Digital Signal Processing units (DSPs), which implement multiplications in dedicated hardware. All the multiplications of the CUT have been mapped in the DSPs, which is the reason why DSP resources are prominent in Table 4.4.

4.5.3 Results of the Fault Emulation Campaign

The fault emulation campaign has been configured to extensively analyze the behavior of the SVM architecture in the presence of faults. The MODNET tool (step ② in Figure 4.2) has identified in the SVM architecture 1350 nodes to be assessed through a transient fault emulation campaign. For each node, a fault has been injected successively for the set of 150 input vectors. It is important to note that the node holds the fault over the entire clock cycle. For each fault emulation, the primary outputs of the SVM architecture have been compared with the correct result (golden), and each observed failure has been logged in the campaign controller.

So, in order to assess the rate of critical failures provoked by the emulation of a single transient fault on a node n of the SVM architecture design, we define the following metric:

$$CriticalFailureRate\ (n) = \frac{\#\ CriticalFailures\ (n)}{\#\ InputVectors} \tag{4.2}$$

in which $\#\ CriticalFailures\ (n)$ is the total number of critical failures provoked by a single transient fault injected at node n of the SVM architecture design, and $\#\ InputVectors$ is the total number of vectors tested at the primary inputs of the SVN architecture with the node n under the same fault emulation.

It is worth mentioning that 58.8% of the faults have been injected on the outputs of the LUTs

connected at the DSP outputs (the Multipliers), while 29.3% of the faults have been injected on the outputs of the LUTs that are in charge of computing the Adders part. The rest of the faults (approximately 11.9%) have been injected on the outputs of the LUTs used by the configuration signals of the SVM architecture. After an extensive fault emulation campaign covering several nodes of the SVM architecture, encompassing a total of 202,500 faults injected in the SVM architecture, results show that more than 29% of emulated single transient faults provoked a critical failure in the SVM architecture. It means that 71% of the fault emulations led to either tolerable failure or no failure. Figure 4.12 presents the histogram of the rates of critical failures (equation (2)) provoked by the emulation of single transient faults on the nodes of the SVM architecture design. The results indicate that most nodes are quite sensitive to transient faults (critical failure rates between 20% and 60%), and only a small number of them has a critical failure rate higher than 60%. Such a considerable critical failure rate on several nodes suggests that the fault masking effects are low due to the parallel configuration of the operators and the short path between the primary inputs and outputs of the SVM architecture. Selective mitigation techniques can be applied to make more robust such nodes with high critical failure rate.

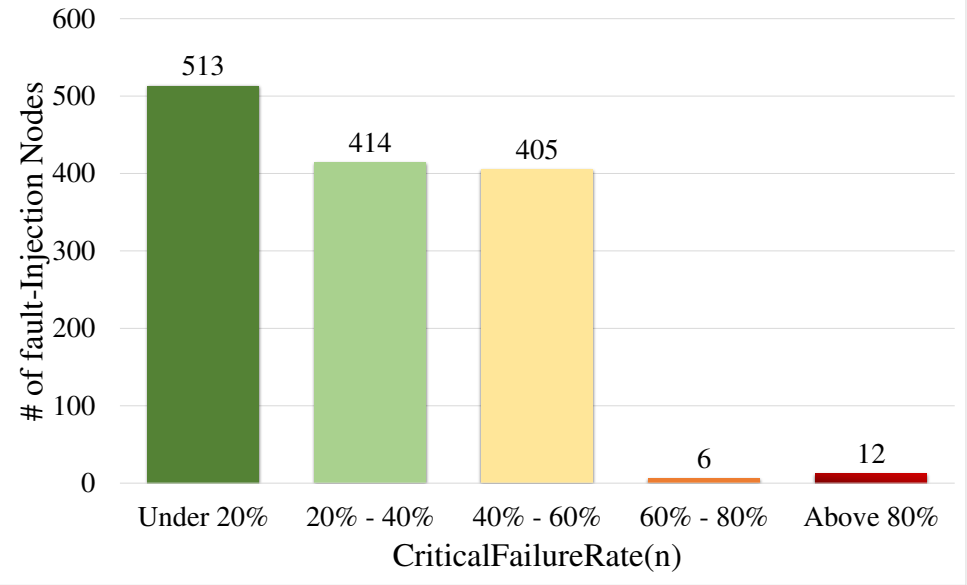

Fig. 4.12: Histogram of the critical failure rate of the injection nodes on the SVM architecture as given by Equation 4.2

Figure 4.13 shows the correlation among the most critical failure rate nodes and their relative position in the SVM's circuitry. It is worth mentioning that those nodes were extracted from the FPGA's LUTs, and their relative positions are logically represented into the Figure 4.11. It means that the SVM architecture is distributed into the FPGAs in basically three logical steps: the Multiplier, the Adder, and the Output. So, the results plotted in the Figure 4.13 indicate that the multiplier is the most sensitive component of the SVM architecture. In this case, all those eight critical components from the multiplier are positioned in the first logical level of the multiplier, i.e., they are LUTs that receive the first computation of the DSPs with most

significant bits. Analyzing the region of the Adders, we can see that eight other faults/nodes can be classified as critical. Unlike the multiplier results, where critical faults are significantly positioned in the first logical level of the SVM architecture, failures in the Adders are diffused in the circuit. It means that the critical failures can occur in different stages of the combinational Adders' circuitry. When the fault injections are performed in Output, only faults happening close or on the actual sign bit cause critical failures. Even though the most susceptible LUTs are spread out in the architecture, they can be mapped. The very small number of LUTs with a high *CriticalFailureRate*, as shown by Figure 4.13, suggests that hardening should be done on these LUTs to improve the SVM fault tolerance with low area overhead.

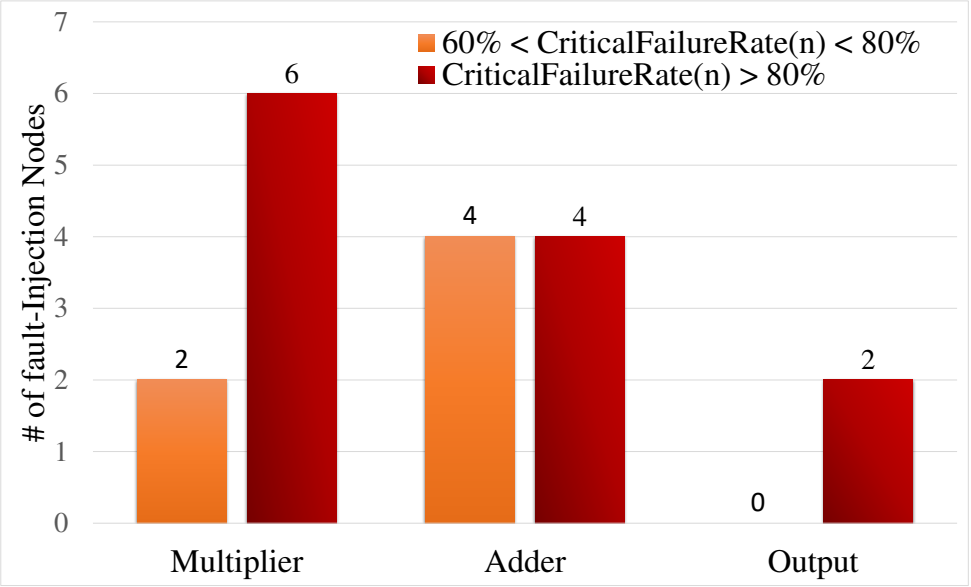

Fig. 4.13: Histogram representing the correlation among the most critical failure rate nodes and their position relative to the the SVM's circuitry implemented in a FPGA.

4.6 Radiation Test Experiment and Results

This section describes the radiation experiments conducted with a neutron source and an analysis of the obtained and assessed results.

4.6.1 Radiation test set-up

A radiation campaign has been performed at the GENEPi2 neutron source, at the *"Laboratoire de Physique Subatomique & Cosmologie"* (LPSC), in Grenoble (France) [118]. The board has been irradiated for 6 hours and 45 minutes, yielding a total neutron fluence of $1.944 * 10^{10}\ n * cm^{-2}$ and average flux of $8 * 10^{5}\ n * cm^{-2}/s$. with an environment temperature of 18°C. Figure 4.14 shows a picture of the test set-up.

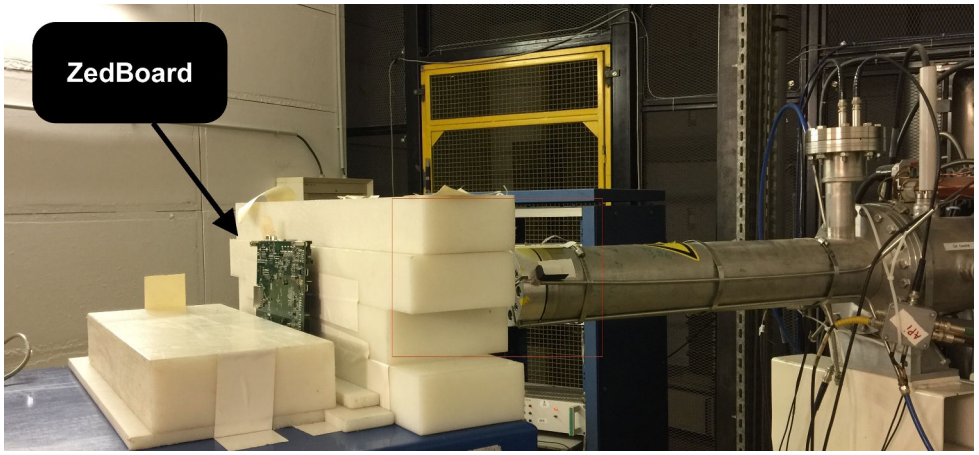

Fig. 4.14: FPGA board installed at the GENEPi2 accelerator neutron facility

The architecture used for the radiation test has a minor difference from the one used in the subsection 4.3.2 and is shown in Figure 4.15. While in the fault emulation campaign the FPGA has been driven by MicroBlaze, in this radiation test experiment we adopted the ARM as controller for the SVM. However, the SVM architecture design on the Programmable Logic (PL) remained unchanged. In order to reduce the amount of input bits sent from the ARM to the SVM architecture, we instantiated a small controller that had 32-bit wide to store the set of input vectors tested on the FPGA. The controller fetches the data from the memory and forwards it to the SVM. It then retrieves the output and expose it through an AXI interconnect.

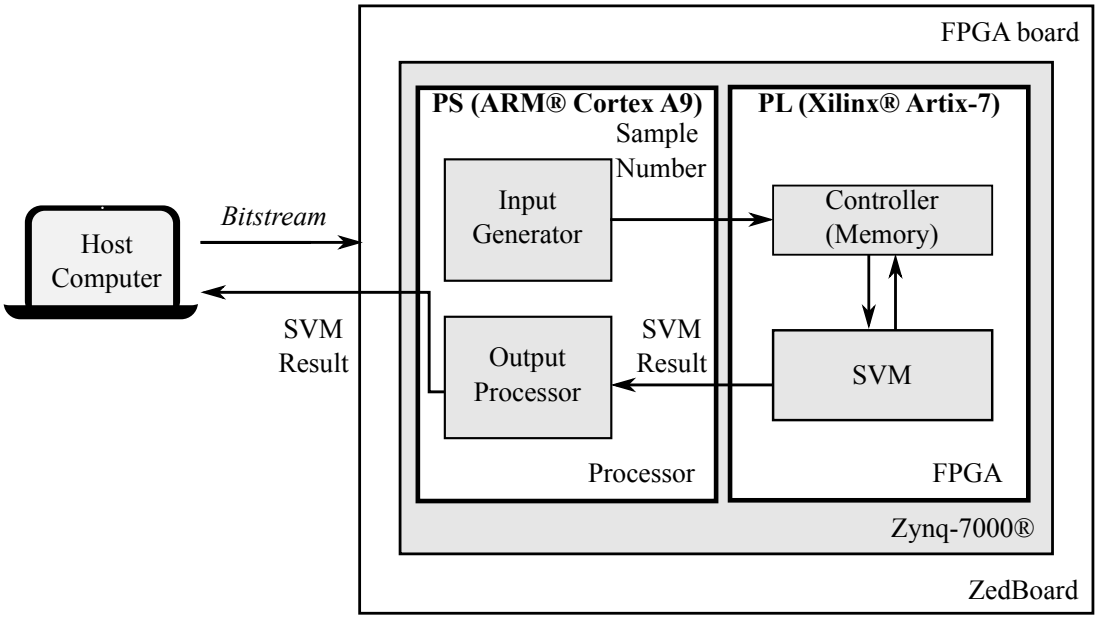

Fig. 4.15: Zynq-7000 set-up under radiation test

The ARM processor provides the controller on the FPGA with the index of the input vector to be tested, as the input vectors are stored in a ROM in the PL, and reads the result. It then forwards the result through serial port. The L2 cache of the ARM processor has been disabled

to reduce the probability of faults affecting the PS [113].

4.6.2 Radiation test method

Figure 4.16 presents the method we have used during the radiation test experiment. The set of input vectors is continuously running in the FPGA. The radiation is able to alter the configuration SRAM, which contains the FPGA bitstream, i.e. it is able to create an error. This error may then lead to an alteration of the SVM calculation structure, mathematically changing the classification function. The score of an input vector evaluated in this faulty SVM may deviate from its expected result, i.e., we analyze the primary outputs of the SVM architecture, and then we compare them with a golden reference. We use this property to identify when an error has happened. Whenever an error has been identified, we rerun the entire set of input vectors on the faulty SVM to classify each input vector according to its type of error, and finally, we log the results. When it is done rerunning the input vectors, the FPGA is reset to erase the error, and the process is restarted.

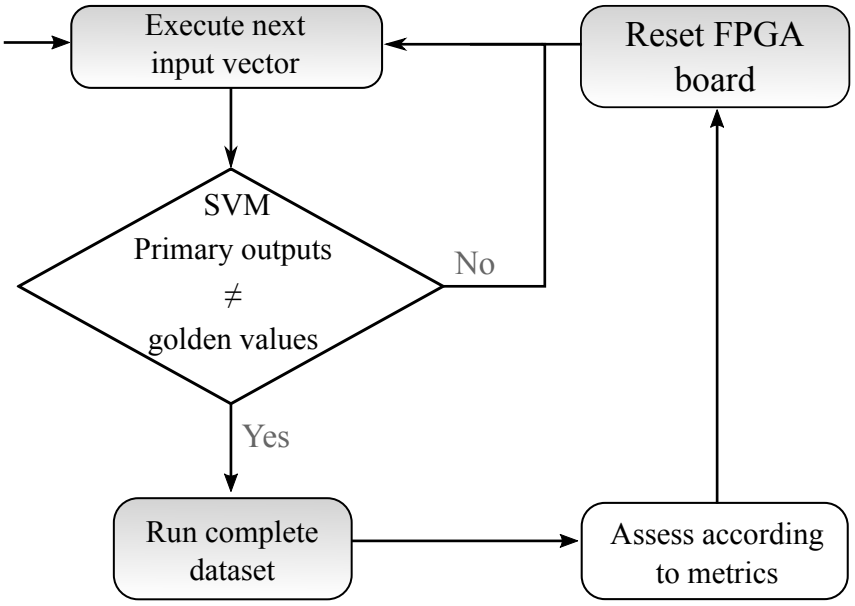

Fig. 4.16: Method used on the radiation test

4.6.3 Assessment of radiation test results

During the neutron radiation test campaign, we have identified 13 errors, of which 2 crashed the FPGA and 11 errors that allowed it to continue to produce results. Even though crashes have been responsible for 15% of the total number of errors, they are not related to the case-study SVM architecture design but due to a fault in the device performing the serial communication with the control computer. The obtained static cross-section and the Failure In Time (FIT) are

respectively $5.65 * 10^{-10}\ cm^2$ and 7.91, considering New York's $14\ n/(cm^2 \cdot h)$ neutron flux at sea-level [83].

A total of 1650 input vectors have been evaluated on SVMs with an altered behavior due to errors, as the 150 input vectors have been input to the SVM when a error have been identified. Of those, 1168 continued to output the expected result (*No Failure*), 92 have been considered as Critical Failure and the remaining 390 as Tolerable Failures. As shown in Figure 4.17, only 5% of the evaluated input vectors have resulted in Critical Failure, thus when there is an error, there is a 95% chance that the error will not lead to a misclassification.

In terms of errors, Figure 4.18 presents a report of the radiation-induced errors that provoked tolerable and critical failures in the SVM architecture during the radiation campaign. 3 of 11 errors caused at least one of the input vectors to be misclassified. This indicates that 27% of the errors identified during the radiation test caused the SVM architecture to produce a critical failure. This result suggests that the case-study SVM architecture indeed has a level of intrinsic fault tolerance, however more information is needed to better identify the nodes that cause critical failures.

On an FPGA-designed SVM architecture, errors may change a x_i, a α or the calculation logic shown in equation (4.1). These reshape and/or dislocate the linear classifier. If there is an error on one of the less significant bits of a x_i or α, the separator displacement may be so small that most of the input vectors are still classified correctly, even though their score, i.e. the output from the classification function, changes.

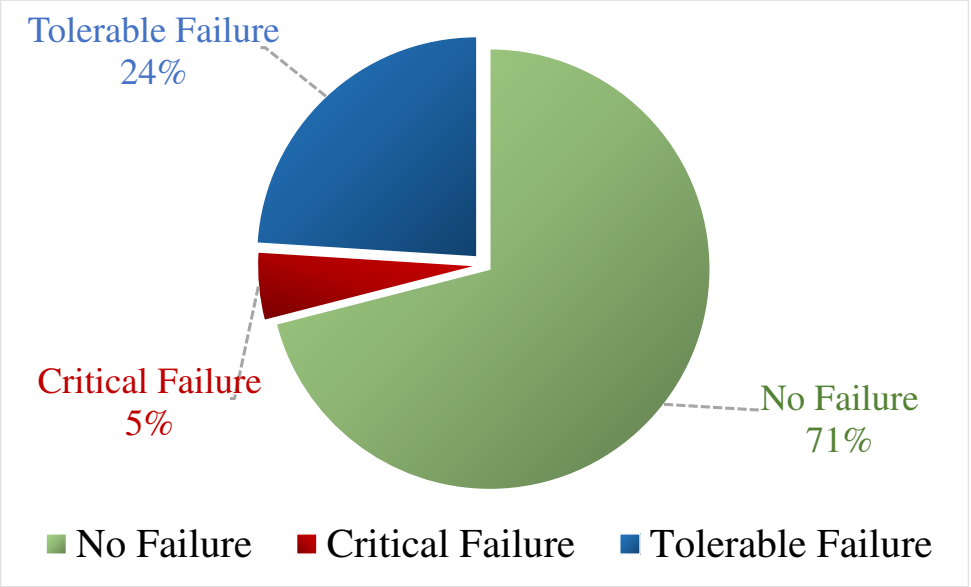

Fig. 4.17: Percentage of failures that have been provoked by 11 neutron radiation-induced errors

The most harmful effects are modifications on the most significant bits of the support vectors $\vec{x_i}$, weights α_i, or changes in the operators (as the technique is implemented in an FPGA). Only one of the 11 non-crashing radiation-induced errors led the majority of the input vectors to be

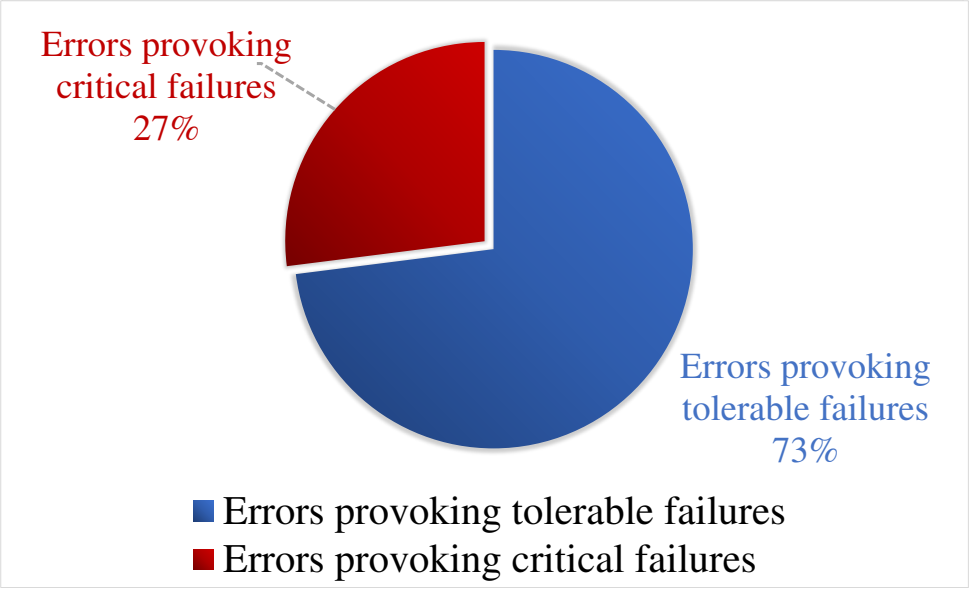

Fig. 4.18: Percentage of neutron radiation-induced errors that provoked 1650 tolerable and critical failures

misclassified (critical failure). This indicates that, for the case-study SVM architecture, this type of error is less likely to happen.

Results from both radiation test and fault emulation campaigns suggest that a hardware-implemented SVM technique has an intrinsic fault tolerance, as identified in other similar experiments with different Machine Learning algorithms [27, 68]. It is worth pointing out that no fault-tolerance mechanisms have been included in the case-study SVM architecture.

4.7 Conclusion

In this Chapter, we proposed NETFI-2 a fault-injection emulation method to study the effects of both SEUs and SETs in digital circuits. Inspired on our existing framework called NETFI, NETFI-2 reduced the hardware overhead of the fault-injection methodology while also allowing for unprecedented flexibility to adapt the CUT model to different SRAM-based FPGA technologies. Furthermore, complex fault-injection campaigns can be easily designed as NETFI-2 relies on a soft-core processor sitting side-by-side with the CUT in an FPGA. It means that NETFI-2 can friendly help the designers to validate a System-On-Chip project design in a very early development stage. Thus, by exploring two appealing cases study based on a Stochastic Bayesian Machine and a Support Vector Machine, we demonstrated that NETFI-2 can conveniently identify weak points in the circuit where SEUs and SETs could occur. Since SET sensible points evidently depend on the final implementation of the combinational part of the circuit, different experiments could be executed for different granularities at the RTL-level. Moreover, by performing fast and configurable fault-injection campaigns, NETFI-2 allowed to detect the sub-

module(s) of the circuit with higher impact in the final error rate, and therefore, which one(s) should be hardened to achieve the expected FPGA or ASIC implementation reliability against radiation effects.

Chapter 5

NoCFI: A Networks-On-Chip Fault Injection Methodology

5.1 Introduction

The increasing complexity of Networks-on-Chip (NoCs) routers and the continuous miniaturization of silicon technology are making this interconnection circuit increasingly vulnerable to transient faults. As we have been discussing through this thesis, it is known that transient faults caused by a single particle strike in sequential elements (i.e., memory cells, latches, and flip-flops) and combinational gates are called *Single Event Upset* (SEU) and *Single Event Transient* (SET), respectively. Furthermore, with the miniaturization of device geometries in nanoscale technologies, it is very likely that a high energy particle strike may affect several adjacent cells in a circuit resulting in *Multiple Bit Upsets* (MBUs) in sequential memory [21] or *Single Event Multiple Transients* (SEMTs) in combinational gates [46]. As a result, the demand for fault injection tools able to accurately emulate multiple failures taking into account the layout of the *Circuit Under Test* (CUT's) circuitry has been increasing in popularity in both academia and industry. Also, a fault injection tool able to identify the most sensitive components on a NoC is crucial to help the designer apply low-cost selective protection schemes.

As described in Chapter 4, *Simulation-based* and *Emulation-based* fault injection are two widely adopted method to test and analyze the effects of transients faults in *Hardware Description Language* (HDL)-based design [94]. In this Chapter, we are interested in Emulation-based fault injection approach that uses the instrumentation-based and FPGA-based techniques [94]. The instrumentation-based technique consists of inserting faults through modification of the *Register Transfer Level* (RTL) model of the circuit instead of using sophisticated reconfiguration features which are not always available in the FPGAs. Therefore, faults are directly injected into the RTL design, which significantly simplifies the injection process, allows the designers to specify which part of the CUT to test, and keeps the HDL of the CUT unchanged. The FPGA-based technique is a methodology used for speeding-up fault injection campaigns. It

can play a useful role in emulating multiple faults on CUTs since the volume of tests increases proportionally with the number of multiples faults.

In this context, many works have proposed fault-injection methods emulated and accelerated by SRAM-based FPGAs. However, to the best of the authors' knowledge, a method that considers the vulnerability to multiple transients faults into the NoCs architecture has not been addressed yet. This gap in the literature motivated us to propose a hybrid method able to emulate single and/or multiples transients faults in the NoCs taking into account the layout position of the circuit's elements. This method is called NoCFI (**N**etworks-**o**n-**C**hip **F**ault **I**njection). NoCFI manipulates the Verilog gate-level netlist provided by the ASIC design flow using the instrumentation-based technique to emulate SET, SEU, SEMT, and MBU in the NoC's logic circuit. Furthermore, NoCFI can perform exhaustive fault-injection campaign in order to identify the most critical cells of NoC's circuitry using either SRAM-based FPGAs or Simulation-based techniques.

5.2 State-of-the-art

Until now, several FPGA-based Networks-on-Chip have been developed [58,65,87,93,97,119]. However, only a few take into consideration either the emulation of transient faults into NoCs using FPGAs [111] or the layout-based techniques to estimate the vulnerability to multiple transients faults caused by single events [13, 40]. For example, in [111] the authors present a fault-injection approach based on a dual-processor system implemented on SRAM-based FPGA which is able to test mesh-based Networks-on-Chip. This approach is based on the physical modification of the FPGA resources in order to insert transient faults through partial and dynamic reconfiguration of the FPGA. Although this approach allows different fault injection partners such as single event upset, stuck-at 0/1 faults, wire faults, bridge and delay faults, it is not able to emulate single event multiple transients faults.

The framework proposed in [84] links fault injection to fault propagation in the floorplan view of a standard cell ASIC. It performs the fault injection campaign by the instrumentation of the gate netlist after place&route and emulation in an FPGA system. Additionally, all the experiment can be controled via an interactive user interface. Although this methodology allows the evaluation of the fault tolerance of a generic circuit taking adjacent cell faults into account, there is no evidence that this framework can be applied in the context of Networks-on-Chips.

The analytical technique presented in [49] and [75] address the Single Event Multiple Transient (SEMT) faults model in a logic circuit. These techniques estimate the vulnerability to transient faults due to SEMTs using only the logic-level netlist for identification of the possible multiple transients faults in the circuit, neglecting the layout-level adjacency circuitry. However, analysis of ISCAS'89 and ITC'99 benchmark circuits reveal that only less than 10% of the netlist adjacent cells are physically adjacent in the layout. Also, more than 60% of physi-

cally adjacent cells are not adjacent in the netlist. It means that the layout of the circuit under test must be taken into consideration for an accurate identification of adjacent cells. In contrast with those two works, the authors in [40] use the SPICE simulation combined with the layout information in order to analyze the sensitive parts of the circuits. This method presents the better accuracy than both the other analytical methods [49, 75]. However, Simulation-based fault injection techniques suffer from a large time simulation.

In [86] the authors have presented a tool named AMUSE which can support the injection of multiple SETs and SEUs. Transients of a selected pulse width can be injected at any time and simultaneously into any combination of circuit nodes. For the affected combinational nodes, the logic value is changed while the injected pulse is active. For sequential nodes, the pulse produces a bit-flip that is kept beyond the end of the pulse until the end of the current clock cycle. In summary, by establishing the concept of collected charge radius, the authors estimated the impact of SEMTs on the sensitivity of an ASIC. However, as the technology scales down, more cells are affected by a single particle strike. As a result, it is difficult to forecast the collected charge radius.

In Chapter 4 we have detailed a methodology to fault injection named NETFI-2. This methodology modifies the Xilinx's libraries as well as the netlist provided by the FPGA-design-flow for injecting faults in the CUT. However, as explained in Chapter 4, NETFI-2 does not take into consideration the layout of the CUT during the fault injection campaign, which means that multiples faults cannot be analyzed. In other words, NETFI-2 is a generic methodology that uses only the FPGA-design flow to select and inject single faults in the CUTs. Although NoCFI was inspired by the NETFI-2 methodology, there are some mainly differences that must be pointed out to help the reader clearly position our contribution:

- **First**, unlike NETFI-2, which modifies the netlist provided by the FPGA-design-flow, NoCFI modifies the gate netlist provided by the ASIC-design-flow process for injecting faults.

- **Second**, since we must perform the place&router to get the netlist we also have the layout position of the circuit's elements. This characteristic permits NoCFI to emulate multiple faults taking into account the relative position of the cells' layout.

- **Third**, we do not need to modify the Xilinx's libraries since our instrumentation is performed only in the gate netlist provided by the place&router. This difference in the instrumentation technique makes the synthesis process more friendly/easily for the user/designer.

- **Fourth**, unlike NETFI-2, which is a generic fault-injection methodology to emulate single faults in SoCs architectures, NoCFI is a more specific methodology able to emulate single and multiple faults in the NoC's architecture.

In summary, some of the limitations presented in the works mentioned above have motivated us to contribute with a solution to emulate single and/or multiple transient faults in NoCs, taking into account the following prerequisites:

1. The layout of the circuit under test must be taken into account for emulating multiple fault injections.
2. The fault injection campaign can be performed either in FPGAs or in HDL-Simulators.
3. The NoC's routing algorithm must be considered during the analyzes of the errors provoked by the fault-injection campaigns.

With those three prerequisites in mind, we proposed an hybrid fault injection method called **NoCFI**, which is described in detail in the remainder of this Chapter.

5.3 2D-NoC Architecture Background

A brief background on the three-dimensional Networks-On-Chip can be found in Chapter 2.3. However, we decided to present in this section a short overview covering the two-dimensional Networks-on-Chip (i.e., 2D-NoC) to better understanding the fault injection methodology proposed in this Chapter.

2D-NoC is a packet-based on-chip interconnection network that can scale its bandwidth proportional to the network size. It overcomes the problems associated with global wire delays and very limited bus bandwidth by replacing ad-hoc shared buses with a modular and flexible interconnect structure comprised of shorter wires thereby increasing simultaneous communication and on-chip bandwidth. A 2D-NoC uses *Routers* to enable a large number of *Processing Element* (PE) to communicate with each other.

Figure 5.1 shows a 4x4 mesh-based NoC where each **PE** is connected to a **Router** by a local interface. The PE can communicate with each other by propagating packets through Routers in the network, and each Router is connected to its neighbors through bidirectional links. Thus, each Router selects a path for the packet to follow until it reaches its destination.

A generic four-stage pipeline 2D-Router architecture is illustrated in Figure 5.2. It consists of a Routing Computation Unit (RCU), a Virtual Channel Allocator (VCA), a Switch Allocator (SW), a Crossbar, and an Input/Output unit. The RCU determines the output port (i.e., the North, South, West, or East) and the virtual channel (i.e. a buffer to storage the flits of a packet) for an incoming packet. Packets can arrive simultaneously in many different input ports of the Router, which may lead to a situation where multiple packets request the same output port and/or virtual channel. However, an output port or a virtual channel should be granted to at most one packet at a time. In this case, the SW grants a packet to access the output port among all requestors while the VCA chooses a packet to get access to the requested Virtual Channel (VC).

Finally, when a packet is granted, the Crossbar unit connects the input port to the corresponding output port which allows the packet previously stored in the buffer (i.e, in the VC) advancing either to next Router or to PE.

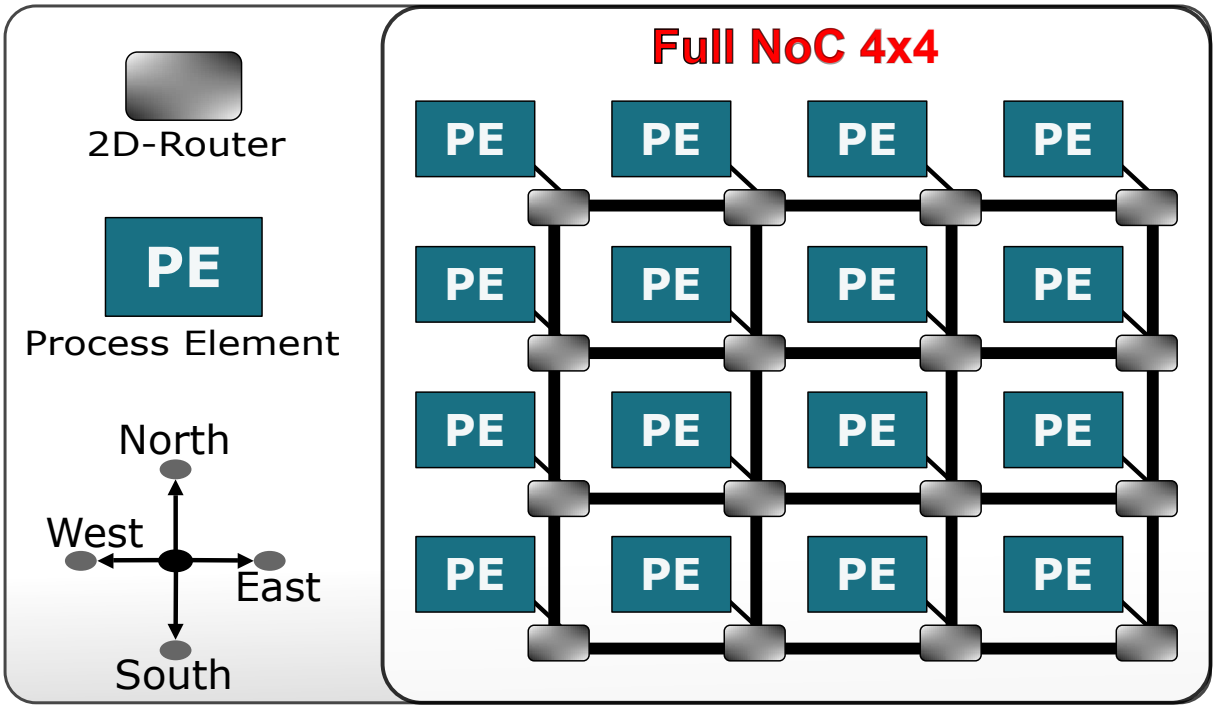

Fig. 5.1: A 4x4 two-dimensional Networks-on-Chip.

5.4 The Effects of Soft-Errors in 2D-Routers

Before explain the NoCFI methodology it is important to describe, in this subsection, the effects of soft errors in the architecture of a 2D-Router. So, a soft error in the routing computation stage would result in the calculation of an inaccurate output port. Since, routing computation stage is the first stage in the routing pipeline, virtual channel allocation and switch allocation stages, which perform allocation based on the output of the routing computation stage, are also affected due to the inaccurate output port. Due to this faulty output port, a packet will be forwarded to an incorrect downstream router, which could potentially result in either packet loss or deadlock or significant increase in the latency of the packet.

A soft error in the virtual channel allocation stage would result in the allocation of an incorrect virtual channel (at the downstream router) to a packet (at the current router). An incorrect virtual channel could be either a non-existing virtual channel or a virtual channel at an output port other than at the required output port or a non-empty virtual channel. The allocation of a non-existing virtual channel or a virtual channel at an output port other than at the required output port would result in a packet drop because; no virtual channel has been allocated to the packet at the correct downstream router. The allocation of a non-empty virtual channel to

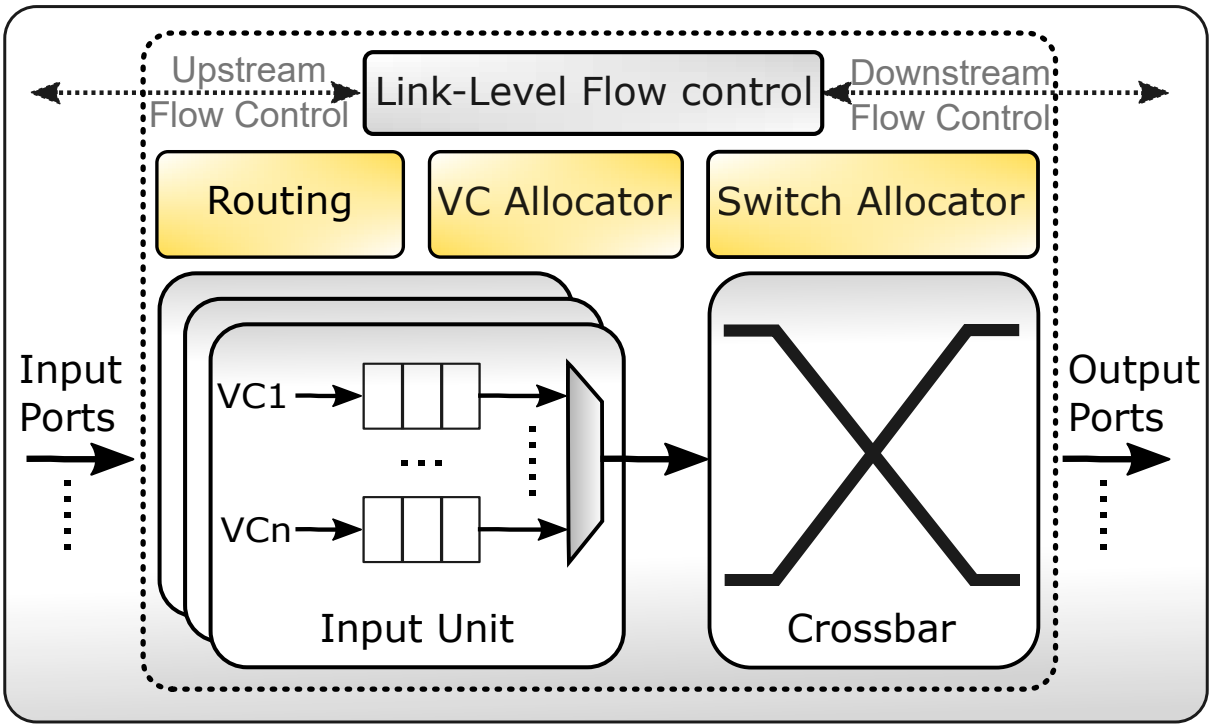

Fig. 5.2: A 4-stage 2D-Router pipeline.

a packet would result in data corruption because the new packet would overwrite the existing packet that is currently occupying the virtual channel. Both packet drop and data corruption call for retransmission of packet which increases packet latency significantly as well as the power consumption.

A soft error in the switch allocation stage could result in flits of the same packet being forwarded to different downstream routers. If the head flit of a packet and a body/tail flit of the same packet are forwarded to different routers, then the body/tail flit will be dropped because there is no virtual channel allocated for them in that specific router. Furthermore, even if the router that has incorrectly received the body/tail flit has an empty virtual channel to store the flit, since there is no routing information present in a body/tail flit, the router cannot route the flit and hence is forced to dropping the flit. As a result, the entire packet needs to be retransmitted that results in significant increase in packet latency.

The crossbar connects the input ports of a router to its output ports. The connections of the crossbar are configured every cycle. A crossbar can be visualized as a group of multiplexers. The number of multiplexers and the size of each multiplexer are determined by the size of the crossbar. From the generic crossbar circuit it can be observed that, to reach a specific output port of the router, flits from any input virtual channel need to traverse through the multiplexer associated with that output port. Since, in the generic crossbar circuit, there is only one path to reach an output port, if a multiplexer is affected by a fault, the path to the associated output port is blocked and hence the port becomes unreachable.

NoC buffers (i.e, the VCs) may be affected by the strike of a charged particle inducing

information errors, compromising the data transmission. In other words, as the packets in a mesh architecture is stored in flits following the head, body, and tail flits, a transient fault in the NoC's buffers can change the packet's information (i.e, the head, body, or tail flits) stored into the buffers. Thus, if a fault is on the head or tail flits, it can change the direction of the packet as well as blocking a buffer causing deadlock. But if the fault is on the body flits, it can change the data information forcing a retransmission of the entire packet.

5.5 NoCFI

In this section, we present our emulation infrastructure for evaluating the impact of multiples faults in the NoC architecture. Also, we described the integration between the ASIC-design flow and the FPGA/Simulation-based flow used in the NoCFI methodology. And finally, we present the fault injection process taking into account the layout information of the 2D-Router as well as its routing algorithm to classify the failures in the NoC.

5.5.1 Methodology

The NoCFI methodology is based on the NETFI-2 described in Chapter 4. However, as detailed in Section 5.2, there are some differences in the approach of each one. So, the main differences is in the use of ASIC-desing flow in order by NoCFI to extract informations about the layout and gate netlist. In this sense, we have adopted a pre-customized cells that represent canonical forms for typical IC functions [106]. Being more specific, we have used standard cell libraries from ST-Microelectronics, which includes low-level logic function such as INVETER, AND, OR, MUX, latches, and flip-flops. This set of cells has a regularized physical structure which allows a more feasible and faster placing and routing and enables an extrapolation of dynamic and static characterization parameters. So, a typical standard cell library contains basically the following main elements [106]:

- Timing and power library definitions: Liberty, Synopsys, DB, IBIS
- Behavioral simulation databases (functional and annotated views): Verilog and VHDL
- Spice simulation databases: CIR, SP, SCS
- Cell circuit views: CDL, DFII
- Cell circuit layouts: GDSII, DFII
- Physical views for placement and routing: LEF

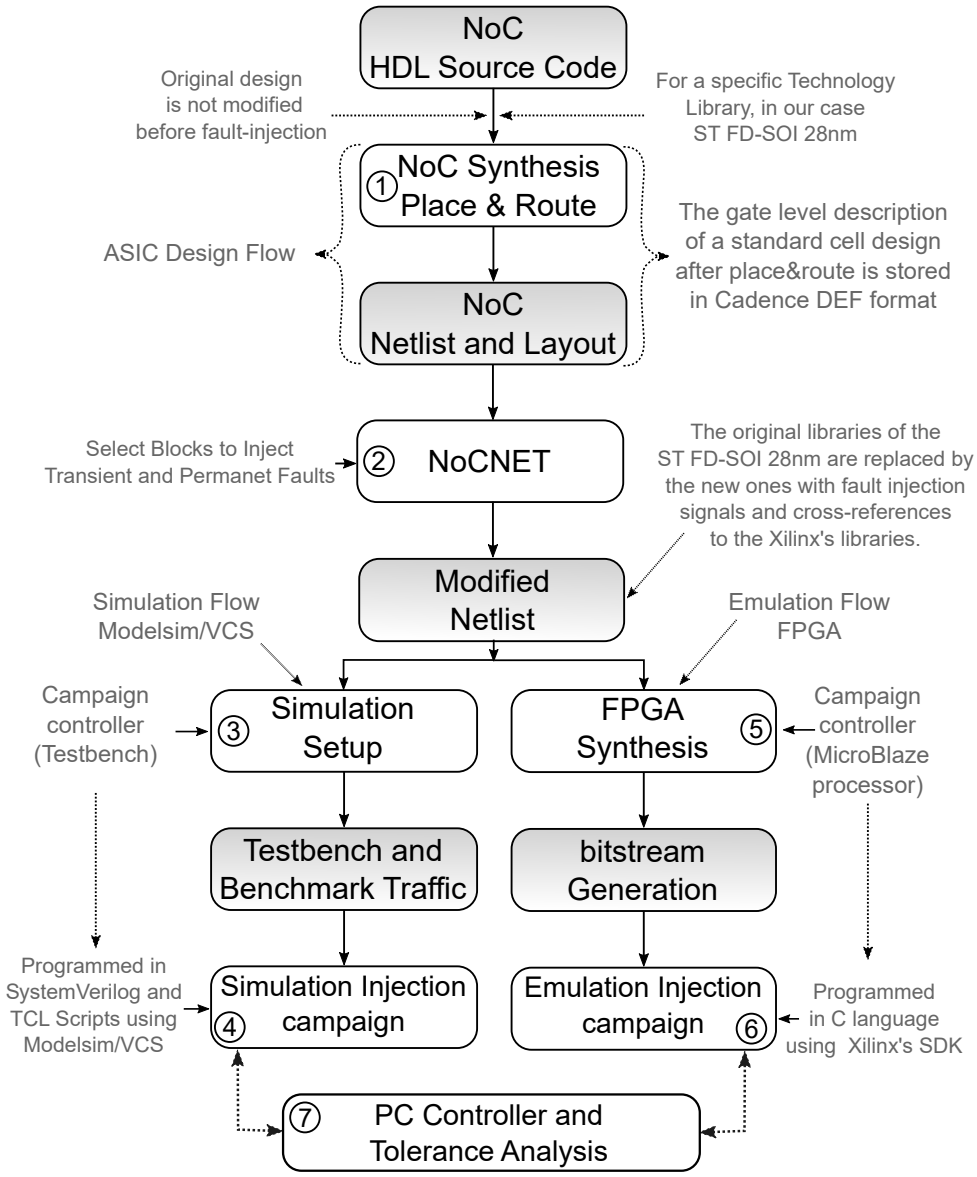

Fig. 5.3: NoCFI work-flow methodology

The NoCFI uses the behavioral simulation databases as weel as the physical views for placement and routing provided by the ASIC-design flow to inject multiples failures. To better understand the NoCFI methodology, we first outline the entire NoCFI methodology in Figure 5.3. In this sense, NoCFI is presented in three main flows (i.e., ASIC-based, FPGA-based, and Simulation-based flow). First, the ASIC-based flow generates the gate netlist and layout information based on the technology adopted. Then, in the FPGA-based flow, all the sensitive parts of the 2D-Router's circuit are identified and classified throughout an exhaustive fault injection campaign. Finally, in the Simulation-based flow, faults are injected in a full 2D-NoC using as a target the most sensitive part of the 2D-Router's circuit previously elected by the FPGA-based flow. In summary, NoCFI methodology can be described in seven steps outlined below.

Initially, in Step ① , the HDL description of the 2D-Router is used for synthesis and then

place&route. The EDA tools used for physical design are Synopsys Design Compiler for synthesis and Cadence SoC Encounter for place&route. The technology adopted was the FD-SOI $28nm$ from ST-Microelectronics. The gate level description of a standard cell design after place&route is the topmost unique description of the future ASIC in terms of abstraction levels. The data to process is placement location of the standard cells stored in Cadence DEF format. This corresponds to the floorplan view. Signal interconnections are stored in Verilog netlist format. The gate netlist and the layout information, are necessary for further steps.

In Step ② , in order to inject faults at each gate output, the **N**etworks-**o**n-**C**hip **NET**list (NoCNET) modify the original gate netlist by means of instrumentation techniques such as one described in Chapter 4. It means that a large number of extra input signals used to access all memory cells and logic blocks of the 2D-Router are inserted in the gate netlist file thus allowing fault injection. The resulting synthesis of the modified gate netlist includes some additional combinational circuitry to the design. Thus, in the case of transient faults, NoCNET modifies all the logic gates of the gate netlist by simply adding an extra multiplexer at the output to select the appropriate value (erroneous or correct). It is worth noting that the original gate netlits is previously obtained in the Step ① using Synopsys Design Compiler and Cadence SoC Encounter.

In summary, to emulate/simulate SEU, SET, MBU, and SEMT in the 2D-Routers some instrumentation in the original gate netlist must be done. As a result, the obtained modified gate netlist can be seen as a different version of the original one, including signals to access the sensitive elements, either to obtain their value or to inject faults. It was adopted a similar instrumentation technique used in the NETFTI-2 and described in subsection 4.3.1. However, it is important to noting that instead of change the Xilinx's library such as NETFI-2, NoCFI modify the built-in library of FD-SOI by adding extra hardware components (i.e., instrumentation technique) like D-flip-flops and related blocks classified into those with enable signal and the those without it. So, the emulation/simulation of SEU, SET, SEMT, and MBU in the 2D-Routers can be briefly described as below:

- **SEU Emulation/Simulation:** in this case, it is required to add some instrumentation hardware around the flip-flops of the design in order to perturb their content at any given moment. To this end, a functionally equivalent structural design of the 2D-Router is obtained (i.e., in terms of gate level primitives) and the hardware inserted around the flip-flops with an injection (*inj*) signal combined with some additional logic (a XOR-gate and a multiplexer) are used to inject faults.

- **SET Emulation/Simulation:** in this case, it is modified all the logic gates of the gate netlist by simply adding an extra multiplexer at the output to select the appropriate value (erroneous or correct).

- **SEMT and MBU Emulation/Simulation:** because the layout information of the circuit

is extracted by means of place&router process, we have the position of a gate and its neighbour gates in the LEF file. Also, we have the correlation between the signal *inj* and the gate from the modified gate netlist. Finally, the instrumentation technique made in the ST-Microelectronics library to emulate/simulate SET and SEU makes it possible to inject multiple faults in the CUT (i.e., 2D-Router) just selecting two or more *inj* signals at the same time.

In Step ③ , a HDL-testbench implemented in SystemVerilog is in charge for managing the fault injection campaign in the NoC's gate netlist modified by NoCNET. In this case, the behavioral simulation databases provided by the library are modified to accept the extra *inj* signals as well as the extra instrumentation hardware that are used for fault injection. It means that the modified netlist obtained in Step ② is attached to the HDL-testbench by using the modified ST FD-SOI $28nm$ library. Also, the traffic benchmark is selected based on the routing algorithm adopted by the 2D-Router.

In Step ④ , the experiment can be directly executed using simulator tools like Modelsim, from Mentor Graphics, or VCS, from Synopsys. The fault-injection campaign is encoded in the HDL-testbench and automated by him. By accessing the local network interfaces as well as the *inj* signal of all 2D-Routers, the HDL-testbench can efficiently execute several iterations of fault-injection experiments randomly selecting the fault points and running different benchmarks traffic.

In Step ⑤ , a campaign controller is integrated within the modified gate netlist for the target FPGA. The campaign controller is implemented in a soft-core processor which is wired throughout AXI interface to the 2D-Router modified by NoCNET. The AXI interface connects all Input/Output ports of the 2D-Router (i.e., North, South, West, East, Local) as well as all signals from Link-level Flow control. Also, all the *inj* signals, which are used to emulate SEU and SET in the 2D-Router, are connected in the campaign controller using AXI interface. And finally, the functional verilog libraries, previously modified to support fault injection, are then attached to the 2D-Router's gate netlist. The resulting *bitstream* implementing the complete circuit (controller, Interface, 2D-Router gate netlist, and FD-SOI libraries) is thus generated and implemented in the target FPGA using Vivado Design Suite from Xilinx.

In Step ⑥ , the FPGA-based experiment can be directly executed from the soft-core processor without requiring additional or external hardware support. Indeed, the whole SEU and SET fault-injection campaign can be conveniently encoded in the processor software. By accessing high-speed interfaces connecting the CUT, the software can efficiently execute several iterations of fault-injection experiments with different traffic benchmark and fault points. In this step, it is possible to execute an exhaustive fault injection campaign since the experiment is performed in FPGAs.

Finally, in Step ⑦ , a Personal Computer (PC) is responsible for collect all the data and analysis the fault injection propagation, the number of errors, and the type of errors. Thus,

each cell in the 2D-Router's circuitry is classified according to the number and type of errors generated. It is important noting that after step ② (NoCNET) this method is partitioned in two indistinctly paths: simulation-flow and emulation-flow. It means that the simulation-flow and the emulation-flow can be used either individually or collaboratively. If it is used collaboratively, the emulation-flow can provides to the simulation-flow some information such as the most critical points or components of failures. Those type of information can accelerate the fault-injection campaign simulation because it can select only the elements of the circuit which are most likely to fail.

5.5.2 NoCFI Architecture

Figure 5.4 shows the presented architecture implementing the NoCFI methodology. NoCFI is integrated into a single FPGA where the 2D-Router and the experiment controller are instantiated and connected by a dedicated interface. To this end, the **A**dvance e**X**tensible **I**nterface (AXI) has been used, since it has been adopted by Xilinx for implementation of complex System-On-Chip (SoC) designs. In this particular case, the MicroBlaze processor (campaign controller) is a master, whereas the 2D-Router is a slave. The MicroBlaze processor can make use of an UART or Ethernet interface to communicate with the outside to report the status of the experiment or its final results.

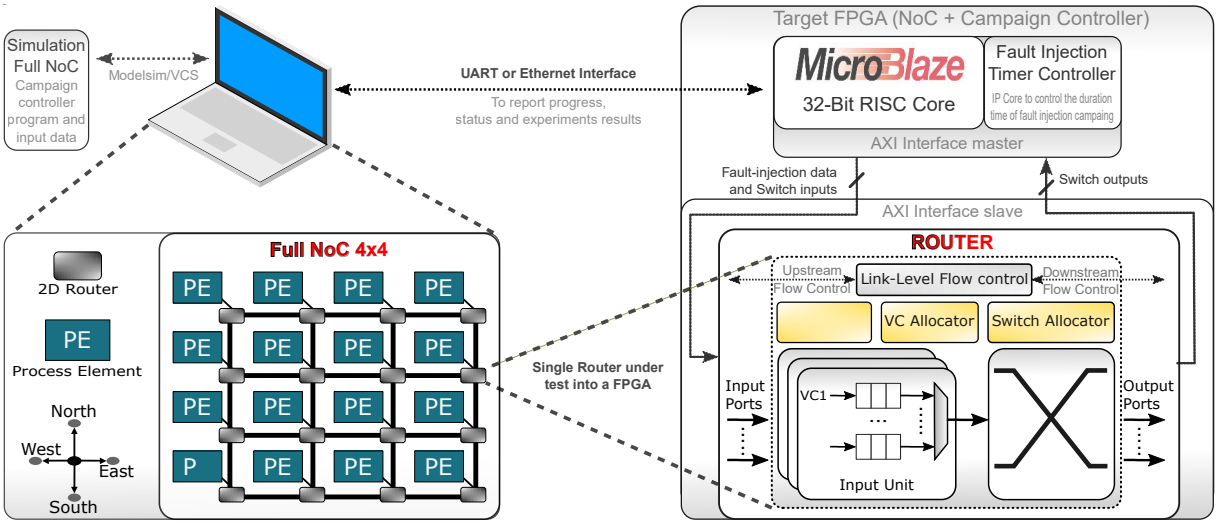

Fig. 5.4: Block diagram of the NoCFI architecture

The AXI interface allows communicating a slave 2D-Router with a master unit (campaign controller) that exchange information while using minimal area in the FPGA. NoCFI uses this communication channel to configure the SEU and SET fault injections in the components already intervened by NoCNET (*inj* signals), as well as to configure the input and read the output values of the 2D-Router. Since the AXI slave device maps the memory addresses of the microprocessor to the 2D-Router fault-injection signals, the fault injection campaign can be conve-

niently defined by software. Furthermore, the clock of the 2D-Router can also be managed via this interface through an IP core named Fault Injection Timer Controller (FITiC). The FITiC can control the clock cycle to accommodate different 2D-Router operating frequencies as well as enabled or disabled the 2D-Router's clock. Also, the FITiC can control the time of the fault injection campaign by means of setting the duration of both SET and SEU as a fraction of the clock cycle.

5.5.3 The Fault Injection Process

The fault injection campaign is performed by a synchronized cooperation between the fault-injection in the cells and the input traffic patterns. This operation is explained in the following four steps:

1. The cell where fault will be injected is selected from a correlation between the signal of *inj* connected to the cell and the physical locations on the Router logic based on layout resources provided by place&route.

2. The MicroBlaze generates all packets patterns to stimulate the Router under test. The source/destination and content data of the packets are generated based on traces of synthetic traffic of a Router in a full 4x4 NoC. Additionally, the MicroBlaze is responsible for checking the flow control of the Router, for injecting faults in the selected cell, and for selecting the type of fault (SEU, SET, MBU, or SEMT). Furthermore, the MicroBlaze is responsible for analyzing the results identifying and classifying the errors.

3. After sending the packet through the Router's input port, the MicroBlaze reads the register of the output ports at each clock cycle. A soft-core, called Fault Injection Timer Controller (FITiC), is responsible for synchronizing the fault injection, the packets patterns injected, and the clock cycle during the fault injection campaign. In other words, the duration of the fault injection can be accurately controlled through the FITiC allowing set the number of valid faults per clock cycle.

4. At the end of the packets patterns stimulus, the correlation between the cell position and the fault signature is saved and sent from MicroBlaze to a PC. The PC will compare, analyze and run the simulation with those information sent by the Microblaze.

These steps are repeated for all the target cells inside the 2D-Router's netlist. During the execution of the fault injection campaign, the processor MicroBlaze communicates to the host-PC by means of the serial/Ethernet connection providing information about the status of the test as well as its final result.

5.6 Evaluation and Validation

We implemented the described approach on a Xilinx Development Board equipped with an Artix-7 XC7A100T-CS324 FPGA. The 2D-Router under test used in this experiment is composed by five input/output ports (North, South, East, West, and Local), two virtual channels (i.e. two buffer per input port), and four pipeline stages as described in Section 5.3. The 2D-Router's HDL-source is an extension of the NoC Netmaker [77]. The processor synthesis and bitstream generation of the FPGA were made using the Xilinx Vivado tool.

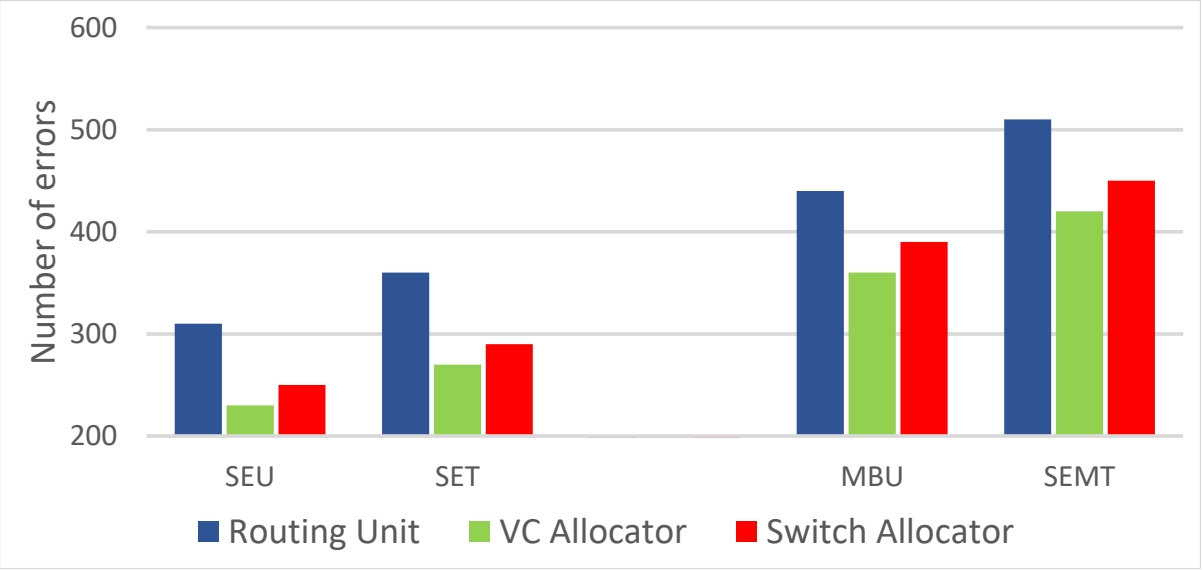

Fig. 5.5: The amount of errors observed in the Router after a fault injection campaign with one fault (SEU/SET) and two faults (MBU/SEMT).

In order to detect errors provoked during the fault injection campaign, the MicroBlaze checks the Router's computation for four differences cases. First, the MicroBlaze verifies the packet's turn taking into account the algorithm used in the routing unit [25]. For example, when the routing unit uses the XY algorithm, the packets must be routed first in the X direction before taking the Y direction. If this order is inverted, the MicroBlaze will report an error. Second, an error is signalized when the VC allocator select a Virtual Channel (i.e., a buffer) which is being used by another packet or belongs to a VN for which the package is forbidden to select. Third, the MicroBlaze will report an error if the Switch Allocator cannot correctly select the compound pair input/output port. Fourth, the MicroBlaze checks either if the packet routed can provoke a deadlock or if the fault injects have blocked the Router. In summary, the Router's computation is considered valid only if it attends for all cases listed above

Figure 5.5 shows the results of the fault injection campaign performed in three elements of the Router: Routing Computation Unit, VC Allocator, and Switch Allocator. The results show that the Routing Computation Unit is the most sensitive elements to transient faults. It has occurred because the Routing Unit is responsible for selecting an output-port as well as a

virtual network which the packet needs to take. Since VC Allocator and Switch Allocator use the output-port and the virtual network provided by the Routing Unit, a mistake in the Routing Computation Unit may be propagated by all subsequent Router's elements.

Table 5.1: Emulation time comparing the fault injection campaign between FPGA-emulation and Gate-level Simulation.

Type	**Platform Test**	
Fault Injection	FPGA-emulation (sec)	Gate-level Simulation (sec)
SET	18	2778
SEMT	34	5280
SEU	12	1908
MBU	25	3960

In order to compare the time saved by the FPGA-based fault injection campaign, we performed a second campaign using the Modelsim simulator, the FD-SOI $28nm$ library modified, and the 2D-Router's gate-level netlist. Table 5.1 shows that our approach can speed up the fault injection campaign in more than *150X* compared with the gate-level Netlist simulation.

5.7 Conclusion

In this Chapter, an hybrid fault injection method for Networks-on-Chip was presented. This method, called NoCFI, combines the FPGA-based emulation with layout-based information allowing an interactive approach to understanding single and multiple transient fault propagation inside the NoC architecture. Compared to an equivalent gate-level fault injection campaign, this method presents a speedup of more than 150 times, which makes possible an exhaustive analysis of complex NoCs. Furthermore, NoCFI can be adapted to test different NoCs' architectures with a minimal modification of its methodology, which makes NoCFI a general method to emulate transients faults in Networks-on-Chip. Besides, since the methodology of NoCFI takes in consideration only the HDL of the Circuit Under Test, it means that this methodology can also be adapted to test others types of HDL-based circuit.

Part IV

CONCLUSIONS

Chapter 6

Conclusions and Perspectives

6.1 Conclusions

Advances in semiconductor technologies and the growing demand for computing power allow implementing more and more embedded processors into the same chip. As a result, Networks-on-Chip are gradually replacing communication buses, which offer more throughput and allow for simplified scaling. At the same time, the shrinking geometric dimensions and the growing circuit complexity lead to an increase in the sensitivity of the circuits to the manufacturing process and their final operating environment. Manufacturing defects and failure rates during the lifetime of the circuit increase when switching from one technology to a more advanced one. Integrating fault tolerance techniques into a circuit becomes essential, especially for circuits which a fault may have critical consequences such as the ones operating in a sensitive environment (i.e., aerospace, automotive, and healthcare). In this sense, this thesis described different techniques for testing, detecting, and correcting transient and permanent failures in the emerging System-on-Chip design. The proposed methods include fault-tolerant techniques against soft-errors for 3D-NoC router architectures, a runtime and fault-tolerant routing scheme for partially connected 3D-NoCs, and fault-injection frameworks for HDL-based designs.

One of the most sensitive parts of networks-on-chip is related to the routing computation unit (RCU). Since the RCU is responsible for selecting an output port and a virtual channel for the packets, a failure in its architecture can provoke critical errors such as deadlock, livelock and/or crash all interconnection logic. In this thesis, soft-errors in the RCU architecture are mitigated by providing efficient and minimalistic mechanisms to detect, mask, and then correct route computation errors. The proposed approach is based on three main assumptions. First, a well-known technique called double-sampling is used to get at different times two samples of the routing computation unit. Then, those two samples are used by the fault-tolerant mechanism in order to detect and/or mask some types of failures. Finally, in the case where failures cannot be masked, a rerouting mechanism is used to repeat the route computation without provoking deadlock. With those three assumptions, we have proved that our approach can detect and cor-

rect misrouting before the packets leave the faulty router. Of course, when a designer makes use of fault-tolerant techniques to increase the circuit reliability, one of the first concerns that comes to mind is the hardware overhead. So, in order to confirm that our approach is feasible in terms of hardware, we have synthesized and compared it with the traditional Triple Modular Redundancy (TMR) technique. Also, simulations were done in order to estimate the performances of the proposed method in the presence of faults as well as in the fault-free operation. The synthesis and simulation results put in evidence the low hardware overhead and the high performance of this approach.

Another challenge in 3D-NoCs is that of transient and permanent faults in the vertical communication of a partially connected 3D topologies. In fact, due to the emergence of TSV as a promising vertical communication technology, it is necessary to provide efficient and reliable routing algorithms for such topologies at a reasonable cost. In Chapter 3, we have demonstrated that it was possible to guarantee 100% of package delivery through a smart attribution of the virtual channels. The proposed routing scheme called FL-RuNS requires the addition of only an asymmetric virtual channel along the West, North, Up, and Down directions that are used as escape path to deliver packets in the presence of faults. It permits the routing algorithm search for a healthy TSV during runtime and/or TSV's failures. Also, since the FL-RuNS uses as a baseline the First-Last routing algorithm, the vertical connection can be placed anywhere in the network. It guarantees full connectivity in the presence of at least one vertical connection in the upward and downward directions. To the best of our knowledge, this algorithm is the first one to allow a complete runtime reconfiguration of the 3D-NoC without dropping packets during transient or permanent faults in the vertical connections. The FL-RuNS algorithm is, therefore, an appealing fault-tolerant solution in terms of cost, performance as well as resilience, making it a great candidate to be adopted in future 3D-NoCs designs.

The second goal of this thesis was the development of a fault injection methodology to estimate the sensitivity to soft-errors of circuits and systems issued from advanced manufacturing technologies. NETFI-2 is an automated method that can be applied to circuits that have their RTL code implemented on an FPGA. It is based on the manipulation of the netlist of the target circuit through the modification of the built-in library of Xilinx in order to emulate SEU and SET faults. One of the advantages of NETFI-2 frameworks is the ability to emulate transient faults in a Circuit Under Test (CUT) using FPGAs. It clearly can accelerate the fault injection campaign hundreds of times when compared with simulation approaches. Also, we have demonstrated that the higher the granularity of the SET emulation is, the more accurate the results will be. In order to evaluate NETFI-2, fault injection campaigns were carried out on two stochastic computer architectures: Bayesian Machine and Support Vector Machine. Using NETFI-2, we have proved that booth architectures are intrinsically resilient to transient faults. And also, we have identified the more sensitive elements for each architecture.

The final contribution of this thesis involved a fault injection methodology called NoCFI,

that can emulate multiple faults in the circuit under test taking into account the layout information. The proposed methodology allows an interactive approach to understanding failure propagation inside the NoCs at an early design stage. NoCFI is a hybrid technique that combines the features of both ASIC-based and FPGA-based design flow in order to emulate transient faults in an NoC. We have demonstrated that, using the Netlist and the files provided by the place&route as well as by modification of the built-in library of the design kit technology, we can emulate multiple failures such as Multiple Bit Upset (MBUs) and Single Event Multiple Transient (SEMTs). A case study using a 4-stage 2D-NoC was used in order to validate the proposed approach. Compared to an equivalent gate-level simulation, this methodology is more than 150 times faster. Also, it is worth noting that this speed gain allows performing exhaustive analysis in the Circuit Under Test (CUT). Finally, we have demonstrated that although this methodology was implemented to study the effects of multiples failures in the NoC's architecture, it can be applied to other CUTs by making only a few adjustments. It makes NoCFI a generic fault injection framework able to emulate multiple faults in any HDL-based design.

6.2 Future Directions

In the field of Networks-on-Chip (NoC), an attractive research direction is the emulation of a full NoC system (cache hierarchy, processing cores, etc..) on FPGAs platforms. It can be interesting since most of the NoC System designs are usually evaluated and validated by means of single-thread cycle-accurate simulation. It means that the vast majority of NoC system simulators are overly slow at performing long-running simulations of full large systems. On the other hand, FPGA-based emulation can provide full parallelism in its execution which clearly provides several orders of magnitude speed-up in comparison with single-thread simulators. However, an important drawback in the FPGA-based emulation is the excessive required time for re-emulation and re-synthesis, when any change is necessary into the NoC. Furthermore, resource limitation is one of the other drawbacks of FPGAs for simulating full NoCs systems. In order to eliminate this obstacle, the virtualization approach has recently been proposed to implement larger NoCs systems on the same FPGA [65,119]. In those cases, synchronization issues should be taken into account when virtualization is employed, which can be a challenge in multi-thread/parallelism systems. So a comprehensive solution for those problems, in an FPGA-based full NoC system emulation, is still an issue that must be addressed.

Yet in the NoC perspective, real time applications for the 3D-NoCs are good candidates for future works due to the fact that it is important to guarantee a fault-tolerant scheme for the router' circuits but also it is important to make the packets able to arrive on-time at their destinations. Furthermore, the works presented in literature have addressed either real-time [57] or fault-tolerant [89,121] in Networks-on-Chip. So, guaranteeing booth fault-tolerance and real-time in a partially connected three-dimensional NoC is still a issue that must be addressed. In

other words, integrating run-time adaptivity and system reconfiguration with fault tolerance and energy-efficient techniques would make significant contributions in this area.

Among different hardware accelerators for neural networks, those methods that use NoC as communication infrastructure give better performance and scalability, as they can better manage the heavy multicast-based inter-neuron and memory-to-neuron traffic [14, 51]. It means that a reconfigurable cluster-based Networks-on-Chip architecture is an excellent candidate for interconnecting future Neural Networks intra-chips. In the field of stochastic computers, the development of complex Machine-Learning Systems-on-Chip (MLSoCs) able to make their own decisions is an attractive subject at the moment. In addition, translating those MLSoCs' algorithms in ASICs designs with low power and area presents a real promise for future works to be developed in the next years.

In the field of fault-injection methodologies, the development of tools able to performed fault injection campaigns in more complex devices requires higher performance in order to carry out a larger fault injection campaign in an acceptable time. Also, the complexity in evaluated Systems-on-Chip with multiple cores is a challenge, since this type of SoCs cannot be tested in a single FPGA due to its larger size and its very complex circuitry. For this reason, we strongly believe that the development of efficient fault-injection methodologies for Many- and Multi-cores SoCs is still an important issue that must be addressed. Thus, an analytical methodology that can emulate/simulate the radiation effect on this type of SoCs can be a promising subject to be investigated.

Our future work includes protecting other critical modules from the NoC router control path such as the virtual channel allocator and switch allocator, offering a comprehensive solution to deal with permanent and transient faults. Also, we have planned to include in ASIC a prototype of a Networks-on-Chip and confront the NoCFI predictions with the results issued from radiation ground testing. Finally, the final version of NoCFI will be provided as open source to allow designers to add any user-specific function, processor, or application.

Bibliography of Author's Publication

This thesis is written based on my research during the period from December 2015 to October 2019 at the University of Grenoble Alpes, France. The following publications are the main references of this thesis:

- ***Journal Publications***

1. A. Coelho, A. Charif, N-E. Zergainoh and R. Velazco, "FL-RuNS: A High Performance and Runtime Reconfigurable Fault-Tolerant Routing Schemes for Partially-Connected 3D Networks-on-Chip," in IEEE Transactions on Nanotechnology, vol. 18, pp. 806-818, 2019. DOI: 10.1109/TNANO.2019.2931271

2. A. Coelho, R. Laurent, M. Solinas M., J. Fraire, E. Mazer, N-E. Zergainoh, R. Velazco, "On the Robustness of Stochastic Bayesian Machines," in IEEE Transactions on Nuclear Science, vol. 64, no. 8, pp. 2276-2283, Aug. 2017. DOI: 10.1109/TNS.2017.2678204

3. M. Trindade, A. Coelho, C. Valadares, R. Vieira, S. Rey, B. Cheymol, M. Balylac, R. Velazco, and R. Bastos. "Assessment of a Hardware-Implemented Machine Learning Technique Under Neutron Irradiation," in IEEE Transactions on Nuclear Science, vol. 66, no. 7, pp. 1441-1448, July 2019. DOI: 10.1109/TNS.2019.2920747

4. A. Charif, A. Coelho, M. Ebrahimi, N. Bagherzadeh and N-E. Zergainoh, "First-Last: A Cost-Effective Adaptive Routing Solution for TSV-Based Three-Dimensional Networks-on-Chip," in IEEE Transactions on Computers, vol. 67, no. 10, pp. 1430-1444, 1 Oct. 2018. DOI: 10.1109/TC.2018.2822269

5. A. Charif, A. Coelho, N-E. Zergainoh and M. Nicolaidis, "A Dynamic Sufficient Condition of Deadlock-Freedom for High-Performance Fault-Tolerant Routing in Networks-on-Chips," in IEEE Transactions on Emerging Topics in Computing. vol. 1, 23 Nov. 2017. DOI: 10.1109/ TETC.2017.2776909

6. A. Charif, A. Coelho, N-E. Zergainoh, M. Nicolaidis, "A Framework for Scalable TSV Assignment and Selection in Three-Dimensional Networks-on-Chips," VLSI Design, vol. 2017, Article ID 9427678, 15 pages, 2017. DOI: https://doi.org/10.1155/2017/9427678.

7. B. Chemli, A. Zitouni, A. Coelho, R. Velazco, "Design of Efficient Pipelined Router Architecture for 3D Network on Chip" in International Journal of Advanced Computer Science and Applications (IJACSA), vol. 8, no. 7, pp 188-194, 7 Jul. 2017. DOI:http://dx.doi.org/10.14569/IJACSA.2017.080725

- ***Conference Publications***

8. A. Coelho, N-E. Zergainoh and R. Velazco, "NoCFI: A Hybrid Fault Injection Method for Networks-on-Chip" IEEE International Latin America Symposium (LATS'19), 4–9 March, 2019, Bolivia, IL, USA.

9. A. Coelho, A. Charif, N-E. Zergainoh and R. Velazco, "An Online Reconfigurable Routing Scheme for Partially Connected 3D Network-On-Chip" IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS'18), 8–10 october, 2018, Chicago, IL, USA.

10. A. Coelho, A. Charif, N.-E. Zergainoh, J. Fraire, R. Velazco, "A Soft-Error Resilient Route Computation Unit for 3D Network-on-Chips", Design, Automation & Test in Europe Conference (DATE'18), 19-23 March, 2018, Dresden, Germany.

11. A. Coelho, M. Solinas, J. Fraire, N.-E. Zergainoh, P. Ferreyra, R. Velazco, "NETFI-2: An Automatic Method for Fault Injection on HDL-Based Designs", Design, Automation & Test in Europe Conference, Univ.Booth (DATE'17), Lausanne, SWITZERLAND, 23-31 march 2017

12. A. Coelho, M. Solinas, R. Laurent, J. Fraire, E. Mazer, N.-E. Zergainoh, S. Karaoui, R. Velazco, "Evidences of Stochastic Bayesian Machines Robustness Against SEUs and SETs", IEEE European Conference on Radiation and its Effects on Components and Systems (RADECS'16), Bremen, GERMANY, 19-23 september 2016

13. M. Solinas, A. Coelho, J. Fraire, N.-E. Zergainoh, R. Velazco, "TGV: Tester Generic and Versatile for radiation effects on advanced VLSI circuits", Automation & Test in Europe Conference, Univ.Booth (DATE'17), Lausanne, SWITZERLAND, 23-31 march 2017

14. A. Charif, N.-E. Zergainoh, A. Coelho, M. Nicolaidis, "Rout3D: A Lightweight Adaptive Routing Algorithm for Tolerating Faulty Vertical Links in 3D-NoCs", 22th IEEE European Test Symposium (ETS'17), pp. 1-6, Limassol, CYPRUS, 22-26 may, 2017.

15. M. Trindade, A. Coelho, C. Valadares, R. Vieira, S. Rey, B. Cheymol, M. Balylac, R. Velazco, and R. Bastos. "Assessment of hardware-implemented support vector machine under radiation effects". In 2018 18th European Conference on Radiation and Its Effects on Components and Systems (RADECS'18), pages 1–5, Gothenburg, Sweden, 16-21 september, 2018.

16. A. Charif, A. Coelho, N.-E. Zergainoh, M. Nicolaidis, "MINI-ESPADA: A Low-Cost Fully Adaptive Routing Mechanism for Networks-on-Chips", IEEE Latin-American Test Symposium (LATS'17), pp. 1-4, Bogota, COLOMBIA, 13-15 march, 2017.

17. T. Bonnoit, A. Coelho, N.-E. Zergainoh, R. Velazco, "SEU Impact in Processor's Control-Unit: Preliminary Results Obtained for LEON3 Soft-Core", 18th IEEE Latin American Test Symposium (LATS'17), pp. 1-4, Bogota, COLOMBIA, 13-15 march, 2017.

18. M. Solinas, A. Coelho, J. Fraire, N.-E. Zergainoh, P. Ferreyra, R. Velazco, "Preliminary Results of NETFI-2: An Automatic Method for Fault Injection on HDL-Based Designs", 18th IEEE Latin-American Test Symposium (LATS'17), Bogota, COLOMBIA, 13-15 march, 2017.

19. A. Charif, A. Coelho, N.-E. Zergainoh, M. Nicolaidis, "Detailed and highly parallelizable cycle-accurate network-on-chip simulation on GPGPU", ACM/IEEE Design Automation Conference (ASP-DAC'17), pp. 672-677, Chiba/Tokyo, JAPAN, 16-19 january, 2017.

20. H. Castro, J. Silveira, A. Coelho, F. Silva, P. Magalhães, O. Lima, "A correction code for multiple cells upsets in memory devices for space applications," 2016 14th IEEE International New Circuits and Systems Conference (NEWCAS'16), Vancouver, BC, 26-29 june, 2016.

References

[1] ACUNHA GUIMARÃES, L., FERREIRA DE PAIVA LEITE, T., POSSAMAI BASTOS, R., AND FESQUET, L. Non-Intrusive Testing Technique for Detection of Trojans in Asynchronous Circuits. In *DATE* (2018).

[2] AHMED, A. B., AND ABDALLAH, A. B. Graceful deadlock-free fault-tolerant routing algorithm for 3d network-on-chip architectures. *Journal of Parallel and Distributed Computing 74*, 4 (2014), 2229 – 2240.

[3] AKBARI, S., SHAFIEE, A., FATHY, M., AND BERANGI, R. AFRA: A low cost high performance reliable routing for 3D mesh NoCs. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (mar 2012), IEEE, pp. 332–337.

[4] ALAGHI, A., AND HAYES, J. Survey of stochastic computing. *ACM Trans. Embed. Comput. Syst. 12*, 2s (May 2013), 92:1–92:19.

[5] ALDERIGHI, M., CASINI, F., D'ANGELO, S., MANCINI, M., CODINACHS, D. M., PASTORE, S., POIVEY, C., SECHI, G. R., SORRENTI, G., AND WEIGAND, R. Experimental validation of fault injection analyses by the flipper tool. *IEEE Transactions on Nuclear Science 57*, 4 (Aug 2010), 2129–2134.

[6] BAHMANI, M., SHEIBANYRAD, A., PÉTROT, F., DUBOIS, F., AND DURANTE, P. A 3D-NoC router implementation exploiting vertically-partially-connected topologies. *Proceedings - 2012 IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2012* (2012), 9–14.

[7] BARAKAT, N., BRADLEY, A. P., AND H. BARAKAT, M. N. Intelligible support vector machines for diagnosis of diabetes mellitus. *IEEE Transactions on Information Technology in Biomedicine 14*, 4 (July 2010), 1114–1120.

[8] BARAZA, J. C., GRACIA, J., BLANC, S., GIL, D., AND GIL, P. J. Enhancement of fault injection techniques based on the modification of vhdl code. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 16*, 6 (June 2008), 693–706.

[9] BAUMANN, R. C. Radiation-induced soft errors in advanced semiconductor technologies. *IEEE Transactions on Device and Materials Reliability 5*, 3 (Sep. 2005), 305–316.

[10] BONCALO, O., AMARICAI, A., SPAGNOL, C., AND POPOVICI, E. Cost effective fpga probabilistic fault emulation. In *2014 NORCHIP* (Oct 2014), pp. 1–4.

[11] BORKAR, S. Thousand core chipsa technology perspective. In *2007 44th ACM/IEEE Design Automation Conference* (June 2007), pp. 746–749.

[12] BURNS, J., MCILRATH, L., KEAST, C., LEWIS, C., LOOMIS, A., WARNER, K., AND WYATT, P. Three-dimensional integrated circuits for low-power, high-bandwidth systems on a chip. In *2001 IEEE International Solid-State Circuits Conference (ISSCC)* (Feb 2001), pp. 268–269.

[13] CAO, X., XIAO, L., LI, J., ZHANG, R., LIU, S., AND WANG, J. A layout-based soft error vulnerability estimation approach for combinational circuits considering single event multiple transients (semts). *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2018), 1–1.

[14] CARRILLO, S., HARKIN, J., MCDAID, L. J., MORGAN, F., PANDE, S., CAWLEY, S., AND MCGINLEY, B. Scalable hierarchical network-on-chip architecture for spiking neural network hardware implementations. *IEEE Transactions on Parallel and Distributed Systems 24*, 12 (Dec 2013), 2451–2461.

[15] CHARIF, A., COELHO, A., EBRAHIMI, M., BAGHERZADEH, N., AND ZERGAINOH, N. E. First-last: A cost-effective adaptive routing solution for tsv-based three-dimensional networks-on-chip. *IEEE Transactions on Computers* (2018), 1–1.

[16] CHARIF, A., COELHO, A., ZERGAINOH, N.-E., AND NICOLAIDIS, M. A Framework for Scalable TSV Assignment and Selection in Three-Dimensional Networks-on-Chips. *VLSI Design 2017* (2017), 1–15.

[17] CHARIF, A., COELHO, A., ZERGAINOH, N. E., AND NICOLAIDIS, M. Detailed and highly parallelizable cycle-accurate network-on-chip simulation on GPGPU. In *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC* (jan 2017), IEEE, pp. 672–677.

[18] CHARIF, A., COELHO, A., ZERGAINOH, N. E., AND NICOLAIDIS, M. A dynamic sufficient condition of deadlock-freedom for high-performance fault-tolerant routing in networks-on-chips. *IEEE Transactions on Emerging Topics in Computing PP*, 99 (2017), 1–1.

[19] CHARIF, A., ZERGAINOH, N.-E., COELHO, A., AND NICOLAIDIS, M. Rout3d: A lightweight adaptive routing algorithm for tolerating faulty vertical links in 3d-nocs. In *22th IEEE European Test Symposium (ETS'17)* (2017), ACM IEEE, pp. 1–6.

[20] CHARIF, A., ZERGAINOH, N. E., AND NICOLAIDIS, M. Addressing transient routing errors in fault-tolerant networks-on-chips. In *2016 21th IEEE European Test Symposium (ETS)* (May 2016), pp. 1–6.

[21] CHATTERJEE, I., NARASIMHAM, B., MAHATME, N. N., BHUVA, B. L., REED, R. A., SCHRIMPF, R. D., WANG, J. K., VEDULA, N., BARTZ, B., AND MONZEL, C. Impact of technology scaling on sram soft error rates. *IEEE Transactions on Nuclear Science 61*, 6 (Dec 2014), 3512–3518.

[22] CHEN, C., AND COTOFANA, S. D. A low cost method to tolerate soft errors in the noc router control plane. In *2013 IEEE International SOC Conference* (Sept 2013), pp. 374–379.

[23] CIVERA, P., MACCHIARULO, L., REBAUDENGO, M., REORDA, M. S., AND VIOLANTE, M. Exploiting fpga-based techniques for fault injection campaigns on vlsi circuits. In *Proceedings 2001 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems* (2001), pp. 250–258.

[24] COELHO, A., CHARIF, A., ZERGAINOH, N., AND VELAZCO, R. A runtime fault-tolerant routing scheme for partially connected 3d networks-on-chip. In *2018 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)* (Oct 2018), pp. 1–6.

[25] COELHO, A., CHARIF, A., ZERGAINOH, N. E., FRAIRE, J., AND VELAZCO, R. A soft-error resilient route computation unit for 3d networks-on-chips. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2018), pp. 1357–1362.

[26] COELHO, A., LAURENT, R., JR, M. S., FRAIRE, J., MAZER, E., ZERGAINOH, N. E., KARAOUI, S., AND VELAZCO, R. On the robustness of stochastic bayesian machines. *IEEE Transactions on Nuclear Science PP*, 99 (2017), 1–1.

[27] COELHO, A., SOLINAS, M., LAURENT, R., FRAIRE, J., MAZER, E., ZERGAINOH, N., KARAOUI, S., AND VELAZCO, R. Evidences of Stochastic Bayesian Machines Robustness Against SEUs and SETs. In *Proceedings of IEEE RADECS'16* (Sept 2016). In Press.

[28] COELHO, A., ZERGAINOH, N., AND VELAZCO, R. Nocfi: A hybrid fault injection method for networks-on-chip. In *2019 IEEE Latin American Test Symposium (LATS)* (March 2019), pp. 1–6.

[29] COELHO, A., ZERGAINOH, N., AND VELAZCO, R. Nocfi: A hybrid fault injection method for networks-on-chip. In *2019 IEEE Latin American Test Symposium (LATS)* (March 2019), pp. 1–6.

[30] CONSTANTINIDES, K., PLAZA, S., BLOME, J., ZHANG, B., BERTACCO, V., MAHLKE, S., AUSTIN, T., AND ORSHANSKY, M. Bulletproof: a defect-tolerant cmp switch architecture. In *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.* (Feb 2006), pp. 5–16.

[31] CRISWELL, T. L., MEASEL, P. R., AND WAHLIN, K. L. Single event upset testing with relativistic heavy ions. *IEEE Transactions on Nuclear Science 31*, 6 (Dec 1984), 1559–1561.

[32] CUNNINGHAM, C. M., AND AVRESKY, D. R. Fault-tolerant adaptive routing for two-dimensional meshes. In *Proceedings of 1995 1st IEEE Symposium on High Performance Computer Architecture* (Jan 1995), pp. 122–131.

[33] DALLY, W., AND TOWLES, B. Route packets, not wires: on-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference* (2001), pp. 684–689.

[34] DANG, K. N., MEYER, M., OKUYAMA, Y., ABDALLAH, A. B., AND TRAN, X.-T. Soft-error resilient 3d network-on-chip router. In *2015 IEEE 7th International Conference on Awareness Science and Technology (iCAST)* (Sept 2015), pp. 84–90.

[35] DAVIS, W. R., WILSON, J., MICK, S., XU, J., HUA, H., MINEO, C., SULE, A. M., STEER, M., AND FRANZON, P. D. Demystifying 3D ICs: The pros and cons of going vertical, jun 2005.

[36] DE ANDRES, D., RUIZ, J. C., GIL, D., AND GIL, P. Fault emulation for dependability evaluation of vlsi systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 16*, 4 (April 2008), 422–431.

[37] DIXIT, A., AND WOOD, A. The impact of new technology on soft error rates. In *Reliability Physics Symposium (IRPS), 2011 IEEE International* (April 2011), pp. 5B.4.1–5B.4.7.

[38] DUARTE, R. P., LOBO, J., FERREIRA, J. F., AND DIAS, J. Synthesis of Bayesian Machines on FPGAs using Stochastic Arithmetic. In *2nd International Workshop on Neuromorphic and Brain-Based Computing Systems (NeuComp 2015), Design Automation Test Europe (DATE2015)* (2015).

[39] DUBOIS, F., SHEIBANYRAD, A., PETROT, F., AND BAHMANI, M. Elevator-First: A Deadlock-Free Distributed Routing Algorithm for Vertically Partially Connected 3D-NoCs. *IEEE Transactions on Computers 62*, 3 (mar 2013), 609–615.

[40] EBRAHIMI, M., ASADI, H., BISHNOI, R., AND TAHOORI, M. B. Layout-based modeling and mitigation of multiple event transients. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 35*, 3 (March 2016), 367–379.

[41] EBRAHIMI, M., AND DANESHTALAB, M. EbDa: A new theory on design and verification of deadlock-free interconnection networks. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)* (June 2017), pp. 703–715.

[42] EBRAHIMI, M., DANESHTALAB, M., LILJEBERG, P., AND TENHUNEN, H. Fault-tolerant method with distributed monitoring and management technique for 3D stacked meshes. In *The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013)* (oct 2013), IEEE, pp. 93–98.

[43] EBRAHIMI, M., DANESHTALAB, M., AND PLOSILA, J. Fault-tolerant routing algorithm for 3d noc using hamiltonian path strategy. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2013), pp. 1601–1604.

[44] EBRAHIMI, M., MOHAMMADI, A., EJLALI, A., AND MIREMADI, S. G. A fast, flexible, and easy-to-develop fpga-based fault injection technique. *Microelectronics Reliability 54*, 5 (2014), 1000 – 1008.

[45] EGHBAL, A., YAGHINI, P. M., BAGHERZADEH, N., AND KHAYAMBASHI, M. Analytical Fault Tolerance Assessment and Metrics for TSV-Based 3D Network-on-Chip. *IEEE Transactions on Computers 64*, 12 (dec 2015), 3591–3604.

[46] EVANS, A., GLORIEUX, M., ALEXANDRESCU, D., POLO, C. B., AND FERLET-CAVROIS, V. Single event multiple transient (semt) measurements in 65 nm bulk technology. In *2016 16th European Conference on Radiation and Its Effects on Components and Systems (RADECS)* (Sept 2016), pp. 1–6.

[47] FAIX, M., MAZER, E., LAURENT, R., ABDALLAH, M. O., HY, R. L., AND LOBO, J. Cognitive computation: A bayesian machine case study. In *Cognitive Informatics Cognitive Computing (ICCI*CC), 2015 IEEE 14th International Conference on* (July 2015), pp. 67–75.

[48] FAURE, F., PERONNARD, P., VELAZCO, R., AND ECOFFET, R. Thesic+: A flexible system for SEE testing. In *Proceedings of Radiation Effects Components and Systems Conference* (Sept. 2002), pp. 231–234.

[49] FAZELI, M., AHMADIAN, S. N., MIREMADI, S. G., ASADI, H., AND TAHOORI, M. B. Soft error rate estimation of digital circuits in the presence of multiple event transients (mets). In *2011 Design, Automation Test in Europe* (March 2011), pp. 1–6.

[50] FEERO, B. S., AND PANDE, P. P. Networks-on-chip in a three-dimensional environment: A performance evaluation. *IEEE Transactions on Computers 58*, 1 (jan 2009), 32–45.

[51] FIRUZAN, A., MODARRESSI, M., DANESHTALAB, M., AND RESHADI, M. Reconfigurable network-on-chip for 3d neural network accelerators. In *2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)* (Oct 2018), pp. 1–8.

[52] FRANK, T., CHAPPAZ, C., LEDUC, P., ARNAUD, L., LORUT, F., MOREAU, S., THUAIRE, A., FARHANE, R. E., AND ANGHEL, L. Resistance increase due to electromigration induced depletion under tsv. In *2011 International Reliability Physics Symposium* (April 2011), pp. 3F.4.1–3F.4.6.

[53] GIUSTI, A., GUZZI, J., CIREŞAN, D. C., HE, F., RODRÍGUEZ, J. P., FONTANA, F., FAESSLER, M., FORSTER, C., SCHMIDHUBER, J., CARO, G. D., SCARAMUZZA, D., AND GAMBARDELLA, L. M. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters 1*, 2 (July 2016), 661–667.

[54] GLASS, C., AND NI, L. The Turn Model for Adaptive Routing. In *Proceedings the 19th Annual International Symposium on Computer Architecture* (New York, NY, USA, 1992), IEEE, pp. 278–287.

[55] GUZMAN-MIRANDA, H., TOMBS, J. N., AND AGUIRRE, M. A. Ft-unshades-up: A platform for the analysis and optimal hardening of embedded systems in radiation environments. In *2008 IEEE International Symposium on Industrial Electronics* (June 2008), pp. 2276–2281.

[56] HARADA, R., MITSUYAMA, Y., HASHIMOTO, M., AND ONOYE, T. Neutron induced single event multiple transients with voltage scaling and body biasing. In *2011 International Reliability Physics Symposium* (April 2011), pp. 3C.4.1–3C.4.5.

[57] HESHAM, S., RETTKOWSKI, J., GOEHRINGER, D., AND ABD EL GHANY, M. A. Survey on real-time networks-on-chip. *IEEE Transactions on Parallel and Distributed Systems 28*, 5 (May 2017), 1500–1517.

[58] HILTON, C., AND NELSON, B. Pnoc: a flexible circuit-switched noc for fpga-based systems. *IEE Proceedings - Computers and Digital Techniques 153*, 3 (May 2006), 181–188.

[59] HOWARD, J., DIGHE, S., VANGAL, S. R., RUHL, G., BORKAR, N., JAIN, S., ERRAGUNTLA, V., KONOW, M., RIEPEN, M., GRIES, M., DROEGE, G., LUND-LARSEN, T., STEIBL, S., BORKAR, S., DE, V. K., AND WIJNGAART, R. V. D. A 48-core ia-32

processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling. *IEEE Journal of Solid-State Circuits 46*, 1 (Jan 2011), 173–183.

[60] HUANG, L., ZHANG, X., EBRAHIMI, M., AND LI, G. Tolerating transient illegal turn faults in nocs. *Microprocess. Microsyst. 43*, C (June 2016), 104–115.

[61] IBE, E., TANIGUCHI, H., YAHAGI, Y., I. SHIMBO, K., AND TOBA, T. Impact of scaling on neutron-induced soft error in srams from a 250 nm to a 22 nm design rule. *IEEE Transactions on Electron Devices 57*, 7 (July 2010), 1527–1538.

[62] IBRAHIM, S. K., AHMED, A., ZEIDAN, M. A. E., AND ZIEDAN, I. E. Machine learning methods for spacecraft telemetry mining. *IEEE Transactions on Aerospace and Electronic Systems 55*, 4 (Aug 2019), 1816–1827.

[63] JEITLER, M., DELVAI, M., AND REICHOR, S. Fuse - a hardware accelerated hdl fault injection tool. In *Programmable Logic, 2009. SPL. 5th Southern Conference on* (April 2009), pp. 89–94.

[64] KALE, V. *Using the MicroBlaze Processor to Accelerate Cost-Sensitive Embedded System Development*. Xilinx, Inc., June 2016.

[65] KAMALI, H. M., AZAR, K. Z., AND HESSABI, S. Ducnoc: A high-throughput fpga-based noc simulator using dual-clock lightweight router micro-architecture. *IEEE Transactions on Computers 67*, 2 (Feb 2018), 208–221.

[66] KAMMLER, D., GUAN, J., ASCHEID, G., LEUPERS, R., AND MEYR, H. A fast and flexible platform for fault injection and evaluation in verilog-based simulations. In *2009 Third IEEE International Conference on Secure Software Integration and Reliability Improvement* (July 2009), pp. 309–314.

[67] KUHN, J. M., SCHWEIZER, T., PETERSON, D., KUHN, T., AND ROSENSTIEL, W. Testing reliability techniques for socs with fault tolerant cgra by using live fpga fault injection. In *2013 International Conference on Field-Programmable Technology (FPT)* (Dec 2013), pp. 462–465.

[68] LIBANO, F., RECH, P., TAMBARA, L., TONFAT, J., AND KASTENSMIDT, F. On the Reliability of Linear Regression and Pattern Recognition Feedforward Artificial Neural Networks in FPGAs. *IEEE Transactions on Nuclear Science 65*, 1 (Jan. 2018), 288–295.

[69] LOI, I., MITRA, S., LEE, T. H., FUJITA, S., AND BENINI, L. A low-overhead fault tolerance scheme for tsv-based 3d network on chip links. In *2008 IEEE/ACM International Conference on Computer-Aided Design* (Nov 2008), pp. 598–602.

[70] LOPEZ-ONGIL, C., GARCIA-VALDERAS, M., PORTELA-GARCIA, M., AND ENTRENA, L. Autonomous fault emulation: A new fpga-based acceleration system for hardness evaluation. *IEEE Transactions on Nuclear Science 54*, 1 (Feb 2007), 252–261.

[71] MAHATME, N. N., JAGANNATHAN, S., LOVELESS, T. D., MASSENGILL, L. W., BHUVA, B. L., WEN, S. ., AND WONG, R. Comparison of combinational and sequential error rates for a deep submicron process. *IEEE Transactions on Nuclear Science 58*, 6 (Dec 2011), 2719–2725.

[72] MANSOUR, W., AND VELAZCO, R. An Automated SEU Fault-Injection Method and Tool for HDL-Based Designs. *IEEE Transactions on Nuclear Science 60*, 4 (Aug 2013), 2728–2733.

[73] MANSOUR, W., AND VELAZCO, R. An automated seu fault-injection method and tool for hdl-based designs. *IEEE Transactions on Nuclear Science 60*, 4 (Aug 2013), 2728–2733.

[74] MANSOUR, W., VELAZCO, R., AYOUBI, R., ZIADE, H., AND FALOU, W. E. A method and an automated tool to perform set fault-injection on hdl-based designs. In *2013 25th International Conference on Microelectronics (ICM)* (Dec 2013), pp. 1–4.

[75] MISKOV-ZIVANOV, N., AND MARCULESCU, D. Multiple transient faults in combinational and sequential circuits: A systematic approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 29*, 10 (Oct 2010), 1614–1627.

[76] MOHAMMADI, A., EBRAHIMI, M., EJLALI, A., AND MIREMADI, S. G. Scfit: A fpga-based fault injection technique for seu fault model. In *2012 Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2012), pp. 586–589.

[77] MULLINS, R. Netmaker, 2009.

[78] MURALI, S., SEICULESCU, C., BENINI, L., AND MICHELI, G. D. Synthesis of networks on chips for 3d systems on chips. In *2009 Asia and South Pacific Design Automation Conference* (Jan 2009), pp. 242–247.

[79] MUROGA, S. *Full-Custom and Semi-Custom Design, The VLSI Handbook, Electrical Engineering Handbook*. CRC Press, 1999.

[80] NAVINER, L., NAVINER, J.-F., DOS SANTOS, G., MARQUES, E., AND PAIVA, N. Fifa: A fault-injection–fault-analysis-based tool for reliability assessment at rtl level. *Microelectronics Reliability 51*, 9 (2011), 1459 – 1463. Proceedings of the 22th European Symposium on the Reliability of Electron Devices, Failure Physics and Analysis.

[81] NIAZMAND, B., AZAD, S. P., FLICH, J., RAIK, J., JERVAN, G., AND HOLLSTEIN, T. Logic-based implementation of fault-tolerant routing in 3D network-on-chips. In *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)* (sep 2016), IEEE, pp. 1–8.

[82] NICOLAIDIS, M. Double-sampling design paradigm - a compendium of architectures. *IEEE Transactions on Device and Materials Reliability 15*, 1 (March 2015), 10–23.

[83] NORMAND, E. Single event upset at ground level. *IEEE Transactions on Nuclear Science 43*, 6 (Dec. 1996), 2742–2750.

[84] NOWOSIELSKI, R., GERLACH, L., BIEBAND, S., PAYÁ-VAYÁ, G., AND BLUME, H. Flint: Layout-oriented fpga-based methodology for fault tolerant asic design. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2015), pp. 297–300.

[85] OWENS, J. D., DALLY, W. J., HO, R., JAYASIMHA, D. N., KECKLER, S. W., AND PEH, L. S. Research challenges for on-chip interconnection networks. *IEEE Micro 27*, 5 (Sept 2007), 96–108.

[86] PAGLIARINI, S., KASTENSMIDT, F., ENTRENA, L., LINDOSO, A., AND MILLAN, E. S. Analyzing the impact of single-event-induced charge sharing in complex circuits. *IEEE Transactions on Nuclear Science 58*, 6 (Dec 2011), 2768–2775.

[87] PARANE, K., M, P. P. B., AND A. B. TALAWAR. Fpga based noc simulation acceleration framework supporting adaptive routing. In *2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)* (March 2018), pp. 1–6.

[88] PARIKH, R., AND BERTACCO, V. Formally enhanced runtime verification to ensure noc functional correctness. In *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (Dec 2011), pp. 410–419.

[89] PARK, D., NICOPOULOS, C., KIM, J., VIJAYKRISHNAN, N., AND DAS, C. R. Exploring fault-tolerant network-on-chip architectures. In *International Conference on Dependable Systems and Networks (DSN'06)* (June 2006), pp. 93–104.

[90] PASRICHA, S., AND ZOU, Y. A low overhead fault tolerant routing scheme for 3D networks-on-chip. *Proceedings of the 12th International Symposium on Quality Electronic Design, ISQED 2011* (2011), 204–211.

[91] PATTI, R. S. Three-dimensional integrated circuits and the future of system-on-chip designs. *Proceedings of the IEEE 94*, 6 (June 2006), 1214–1224.

[92] POURNAGHDALI, F., RAJABZADEH, A., AND AHMADI, M. Vhdlsfi: A simulation-based multi-bit fault injection for dependability analysis. In *ICCKE 2013* (Oct 2013), pp. 354–360.

[93] PRABHU PRASAD, B. M., PARANE, K., AND TALAWAR, B. High-performance noc simulation acceleration framework employing the xilinx dsp48e1 blocks. In *2019 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)* (April 2019), pp. 1–4.

[94] QUINN, H., AND WIRTHLIN, M. Validation techniques for fault emulation of sram-based fpgas. *IEEE Transactions on Nuclear Science 62*, 4 (Aug 2015), 1487–1500.

[95] QUINN, H. M., BLACK, D. A., ROBINSON, W. H., AND BUCHNER, S. P. Fault simulation and emulation tools to augment radiation-hardness assurance testing. *IEEE Transactions on Nuclear Science 60*, 3 (June 2013), 2119–2142.

[96] RAMSAY, F. R. Automation of design for uncommitted logic array. In *17th Design Automation Conference* (June 1980), pp. 100–107.

[97] RETTKOWSKI, J., AND GÖHRINGER, D. Rar-noc: A reconfigurable and adaptive routable network-on-chip for fpga-based multiprocessor systems. In *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)* (Dec 2014), pp. 1–6.

[98] RIJPKEMA, E., GOOSSENS, K. G. W., RADULESCU, A., DIELISSEN, J., VAN MEERBERGEN, J., WIELAGE, P., AND WATERLANDER, E. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *2003 Design, Automation and Test in Europe Conference and Exhibition* (March 2003), pp. 350–355.

[99] RUIZ-LLATA, M., GUARNIZO, G., AND YÉBENES-CALVINO, M. FPGA implementation of a support vector machine for classification and regression. In *The 2010 International Joint Conference on Neural Networks (IJCNN)* (July 2010), pp. 1–5.

[100] SALAMAT, R., EBRAHIMI, M., BAGHERZADEH, N., AND VERBEEK, F. CoBRA: Low cost compensation of TSV failures in 3D-NoC. In *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)* (sep 2016), IEEE, pp. 115–120.

[101] SALAMAT, R., KHAYAMBASHI, M., EBRAHIMI, M., AND BAGERZADEH, N. Lead: An adaptive 3d-noc routing algorithm with queuing-theory based analytical verification. *IEEE Transactions on Computers PP*, 99 (2018), 1–1.

[102] SALAMAT, R., KHAYAMBASHI, M., EBRAHIMI, M., AND BAGHERZADEH, N. A Resilient Routing Algorithm with Formal Reliability Analysis for Partially Connected 3D-NoCs. *IEEE Transactions on Computers 13*, 9 (2016), 1–1.

[103] SCHONWALD, T., ZIMMERMANN, J., BRINGMANN, O., AND ROSENSTIEL, W. Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures. In *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)* (Aug 2007), pp. 527–534.

[104] SERAFY, C., AND SRIVASTAVA, A. Online tsv health monitoring and built-in self-repair to overcome aging. In *2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)* (Oct 2013), pp. 224–229.

[105] SERRANO, F., CLEMENTE, J. A., AND MECHA, H. A methodology to emulate single event upsets in flip-flops using fpgas through partial reconfiguration and instrumentation. *IEEE Transactions on Nuclear Science 62*, 4 (2015), 1617–1624.

[106] SHARMA, R. *Characterization and Modeling of Digital Circuits*. Paripath.com, 11 2015.

[107] SHENG, W., XIAO, L., AND MAO, Z. A novel soft error sensitivity characterization technique based on simulated fault injection and constrained association analysis. In *2008 15th IEEE International Conference on Electronics, Circuits and Systems* (Aug 2008), pp. 766–769.

[108] SHIVAKUMAR, P., KISTLER, M., KECKLER, S. W., BURGER, D., AND ALVISI, L. Modeling the effect of technology trends on the soft error rate of combinational logic. In *Proceedings International Conference on Dependable Systems and Networks* (2002), pp. 389–398.

[109] SHOKROLAH-SHIRAZI, M., AND MIREMADI, S. G. Fpga-based fault injection into synthesizable verilog hdl models. In *Secure System Integration and Reliability Improvement, 2008. SSIRI '08. Second International Conference on* (July 2008), pp. 143–149.

[110] SOLINAS, M., COELHO, A., FRAIRE, J. A., ZERGAINOH, N. E., FERREYRA, P. A., AND VELAZCO, R. Preliminary results of netfi-2: An automatic method for fault injection on hdl-based designs. In *2017 18th IEEE Latin American Test Symposium (LATS)* (March 2017), pp. 1–4.

[111] STERPONE, L., SABENA, D., AND REORDA, M. S. A new fault injection approach for testing network-on-chips. In *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing* (Feb 2012), pp. 530–535.

[112] TAHERI, E., ISAKOV, M., PATOOGHY, A., AND KINSY, M. A. Advertiser elevator: A fault tolerant routing algorithm for partially connected 3d network-on-chips. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)* (Aug 2017), pp. 136–139.

[113] TAMBARA, L. A., RECH, P., CHIELLE, E., TONFAT, J., AND KASTENSMIDT, F. L. Analyzing the Impact of Radiation-Induced Failures in Programmable SoCs. *IEEE Transactions on Nuclear Science 63*, 4 (Aug. 2016), 2217–2224.

[114] TARRILLO, J., KASTENSMIDT, F. L., RECH, P., FROST, C., AND VALDERRAMA, C. Neutron cross-section of n-modular redundancy technique in sram-based fpgas. *IEEE Transactions on Nuclear Science 61*, 4 (Aug 2014), 1558–1566.

[115] TUZOV, I., RUIZ, J. C., ANDRÉS, D. D., AND GIL, P. Speeding-up simulation-based fault injection of complex hdl models. In *2016 Seventh Latin-American Symposium on Dependable Computing (LADC)* (Oct 2016), pp. 51–60.

[116] VAPNIK, V. *The Nature of Statistical Learning Theory*. Springer Science & Business Media, June 2013. Google-Books-ID: EqgACAAAQBAJ.

[117] VELAZCO, R., FOUCARD, G., AND PERONNARD, P. Combining results of accelerated radiation tests and fault injections to predict the error rate of an application implemented in sram-based fpgas. *IEEE Transactions on Nuclear Science 57*, 6 (Dec 2010), 3500–3505.

[118] VILLA, F., BAYLAC, M., REY, S., ROSSETTO, O., MANSOUR, W., RAMOS, P., VELAZCO, R., AND HUBERT, G. Accelerator-Based Neutron Irradiation of Integrated Circuits at GENEPI2 (France). In *2014 IEEE Radiation Effects Data Workshop (REDW)* (July 2014), pp. 1–5.

[119] WANG, D., LO, C., VASILJEVIC, J., ENRIGHT JERGER, N., AND GREGORY STEFFAN, J. Dart: A programmable architecture for noc simulation on fpgas. *IEEE Transactions on Computers 63*, 3 (March 2014), 664–678.

[120] WANG, J. C., LIAN, L. X., LIN, Y. Y., AND ZHAO, J. H. VLSI Design for SVM-Based Speaker Verification System. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 23*, 7 (July 2015), 1355–1359.

[121] WERNER, S., NAVARIDAS, J., AND LUJÁN, M. A survey on design approaches to circumvent permanent faults in networks-on-chip. *ACM Comput. Surv. 48*, 4 (Mar. 2016), 59:1–59:36.

[122] XILINX. Zynq-7000 All Programmable SoC.

[123] XILINX. Axi reference guide v13.4, 2012.

[124] YE, F., FIROUZI, F., YANG, Y., CHAKRABARTY, K., AND TAHOORI, M. B. On-Chip Droop-Induced Circuit Delay Prediction Based on Support-Vector Machines. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 35*, 4 (Apr. 2016), 665–678.

[125] YU, Q., AND AMPADU, P. A dual-layer method for transient and permanent error co-management in noc links. *IEEE Transactions on Circuits and Systems II: Express Briefs 58*, 1 (Jan 2011), 36–40.

[126] ZHANG, T., ZHAN, Y., AND SAPATNEKAR, S. S. Temperature-aware routing in 3d ics. In *Asia and South Pacific Conference on Design Automation, 2006.* (Jan 2006), pp. 6–11.

[127] ZHANG, X., EBRAHIMI, M., HUANG, L., AND LI, G. Fault-resilient routing unit in nocs. In *2015 28th IEEE International System-on-Chip Conference (SOCC)* (Sept 2015), pp. 164–169.

[128] ZHANG, X., EBRAHIMI, M., HUANG, L., LI, G., AND JANTSCH, A. A routing-level solution for fault detection, masking, and tolerance in nocs. In *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing* (March 2015), pp. 365–369.

Titre : Tolérance aux fautes et fiabilité pour les réseaux sur puce 3D partiellement connectés

Résumé : Les avantages combinés des circuits intégrés 3D et des NoCs offrent la possibilité de concevoir un système haute performance dans une zone limitée de la puce. Les NoCs 3D souffrent de certains problèmes de fiabilité tels que la variabilité des processus de fabrication 3D-IC. En particulier, le faible rendement de la connexion verticale a un impact significatif sur la conception des piles de matrices tridimensionnelles avec un grand nombre de TSV. De même, les progrès des technologies de fabrication de circuits intégrés entraînent une augmentation potentielle de leur sensibilité aux effets des rayonnements présents dans l'environnement dans lequel ils vont fonctionner. En fait, le nombre croissant de défaillances transitoires est devenu, au cours des dernières années, une préoccupation majeure dans la conception des systèmes de contrôle critiques. Par conséquent, l'évaluation de la sensibilité des circuits et des applications aux événements causés par les particules énergétiques présentes dans l'environnement réel est une préoccupation majeure à laquelle il faut répondre. Cette thèse présente donc des contributions dans deux domaines importants de la recherche sur la fiabilité : dans la conception et la mise en œuvre de schémas de routage à tolérance de pannes sans blocage pour les réseaux sur puce tridimensionnels émergents ; et dans la conception de cadres d'injection de défauts capables d'émuler des défauts transitoires simples et multiples dans les circuits basés sur HDL. La première partie de cette thèse aborde les problèmes des défauts transitoires et permanents dans l'architecture des NoCs 3D et présente une nouvelle unité de calcul de routage résiliente ainsi qu'un nouveau schéma de routage tolérant aux défauts d'exécution. Un nouveau mécanisme résilient est introduit afin de tolérer les défauts transitoires se produisant dans l'unité de calcul de route (RCU), qui est l'élément logique le plus important dans les routeurs NoC. En combinant un circuit de détection de défauts fiable à double échantillonnage au niveau du circuit et un mécanisme de réacheminement économique, nous développons une solution complète de tolérance aux fautes qui peut détecter et corriger efficacement ces erreurs fatales avant que les paquets affectés ne quittent le routeur. Pourtant, dans la première partie de cette thèse, un nouveau schéma de routage à tolérance de pannes pour les réseaux 3D sur puce à connexion verticale partielle appelé FL-RuNS est présenté. Grâce à une distribution asymétrique des canaux virtuels, FL-RuNS peut garantir une distribution de paquets à 100% sous un ensemble non contraint de temps d'exécution et de pannes permanentes des liaisons verticales. Dans le but d'émuler les effets du rayonnement sur les nouvelles conceptions de SoCs, la deuxième partie de cette thèse aborde les méthodologies d'injection de fautes en introduisant deux outils appelés NETFI-2 et NoCFI. NETFI-2 est une méthodologie d'injection de fautes capable d'émuler des défauts transitoires tels que SEU et SET dans un circuit HDL. Enfin, dans la dernière partie de ce travail, nous présentons NoCFI comme une nouvelle méthodologie pour injecter des défauts multiples tels que les MBU et SEMT dans une architecture de réseaux sur puce.

Mots-clés : Réseaux tridimensionnels sur puce, erreurs logicielles, algorithme de routage tolérant aux pannes, pannes transitoires et permanentes

Title: Fault Tolerance and Reliability for Partially Connected 3D Networks-on-Chip

Abstract: The combined benefits of 3D IC and Networks-on-Chip (NoC) schemes provide the possibility of designing a high-performance system in a limited chip area. The major advantages of Three-Dimensional Networks-on-Chip (3D-NoCs) are a considerable reduction in the average wire length and wire delay, resulting in lower power consumption and higher performance. However, 3D-NoCs suffer from some reliability issues such as the process variability of 3D-IC manufacturing. In particular, the low yield of vertical connection significantly impacts the design of three-dimensional die stacks with a large number of Through Silicon Via (TSV). Equally concerning, advances in integrated circuit manufacturing technologies are resulting in a potential increase in their sensitivity to the effects of radiation present in the environment in which they will operate. In fact, the increasing number of transient faults has become, in recent years, a major concern in the design of critical SoC. As a result, the evaluation of the sensitivity of circuits and applications to events caused by energetic particles present in the real environment is a major concern that needs to be addressed. So, this thesis presents contributions in two important areas of reliability research: in the design and implementation of deadlock-free fault-tolerant routing schemes for the emerging three-dimensional Networks-on-Chips; and in the design of fault injection frameworks able to emulate single and multiple transient faults in the HDL-based circuits. The first part of this thesis addresses the issues of transient and permanent faults in the architecture of 3D-NoCs and introduces a new resilient routing computation unit as well as a new runtime fault-tolerant routing scheme. A novel resilient mechanism is introduced in order to tolerate transient faults occurring in the route computation unit (RCU), which is the most important logical element in NoC routers. Failures in the RCU can provoke misrouting, which may lead to severe effects such as deadlocks or packet loss, corrupting the operation of the entire chip. By combining a reliable fault detection circuit leveraging circuit-level double-sampling, with a cost-effective rerouting mechanism, we develop a full fault-tolerance solution that can efficiently detect and correct such fatal errors before the affected packets leave the router. Yet in the first part of this thesis, a novel fault-tolerant routing scheme for vertically-partially-connected 3D Networks-on-Chip called FL-RuNS is presented. Thanks to an asymmetric distribution of virtual channels, FL-RuNS can guarantee 100% packet delivery under an unconstrained set of runtime and permanent vertical link failures. With the aim to emulate the radiation effects on new SoCs designs, the second part of this thesis addresses the fault injection methodologies by introducing two frameworks named NETFI-2 (Netlist Fault Injection) and NoCFI (Networks-on-Chip Fault Injection). NETFI-2 is a fault injection methodology able to emulate transient faults such as Single Event Upsets (SEU) and Single Event Transient (SET) in a HDL-based design. Finally, in the last part of this work, we present NoCFI as a novel methodology to inject multiple faults such as MBUs and SEMT in a Networks-on-Chip architecture.

Keywords: Three-dimensional Networks-on-Chip, Soft-errors, Fault-tolerant routing algorithm, transient and permanent faults

Thèse préparée au Laboratoire TIMA / Thesis prepared at TIMA Laboratory
Techniques de l'Informatique et de la Microélectronique pour l'Architecture des ordinateurs
Techniques of Informatics and Microelectronics for integrated systems Architecture
46 avenue Félix Viallet – 38031 GRENOBLE Cedex - France

ISBN: 978-2-11-129261-1