HAL open science

Tolerating Errors in NoC: A Lightweight Region-Based Fault-Mitigation Method

Romain Mercier, Cédric Killian, Angeliki Kritikakou, Youri Helen, Daniel Chillet

To cite this version:

Romain Mercier, Cédric Killian, Angeliki Kritikakou, Youri Helen, Daniel Chillet. Tolerating Errors in NoC: A Lightweight Region-Based Fault-Mitigation Method. SELSE 2022 - IEEE Workshop on Silicon Errors in Logic – System Effects, May 2022, [Virtual], France. pp.1-7. hal-03926148

HAL Id: hal-03926148

https://inria.hal.science/hal-03926148

Submitted on 6 Jan 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Tolerating Errors in NoC: A Lightweight Region-Based Fault-Mitigation Method

† Romain Mercier, † Cédric Killian, ‡ Angeliki Kritikakou, § Youri Helen, and † Daniel Chillet
† ‡ *Univ Rennes, Inria, CNRS, IRISA* – § *DGA MI*
† Lannion, France – ‡ § Rennes, France
† ‡ {first-name}.{last-name}@irisa.fr – § youri.helen@intradef.gouv.fr

***Abstract*—Due to transistor shrinking and core number increasing in System-on-Chip (SoC), fault tolerance has become essential. Faults occurring to Network-on-Chips (NoCs) of those systems have a significant impact, due to the high amount of data crossing the NoC for communication. However, existing fault correction approaches cannot efficiently address several permanent faults on NoC NoC, due to their high hardware costs. To mitigate the impact of faults, existing works shuffle the bits inside a flit, transferring the impact of faults on the least significant bits. However, such approaches are applied at a fine-grained level, providing fault mitigation efficiency but with significant hardware costs. To address this limitation, this work proposes a region-based bit-shuffling technique, applied at a coarse-grain level, that trades off fault mitigation efficiency in order to save hardware costs.**

***Index Terms*—Network-on-Chip, Fault Mitigation, Approximate Computing, Hardware Costs Saving, Bit-Shuffling, BiSu, R-BiSu**

I. Introduction

Since several decades, technology evolution enabled high transistor density reaching today billions of transistors per chip. While frequencies and chip density met the power limit, performance increase has been reached by adding more cores into a single chip, giving birth to System-on-Chips (SoCs). However, the increase in number of cores induced more and more data transfers, which cannot be handled by conventional communication means. To address this gap, Network-on-Chip (NoC) appeared as a scalable solution to manage communications between a large number of cores [1].

In this technology era, NoC became more sensitive to permanent faults. As the transistor size reaches 10 nm and below [2], the engraving thinness causes more and more hardware defects, due to manufacturing process [3]. During system operation, aging defects [4], like electromigration, Negative-Bias Temperature Instability (NBTI), Hot Carrier Injection (HCI) and Time-Dependent Dielectric Breakdown (TDDB), and radiations [5] become additional sources of permanent faults. Manufacturing and aging defects are well known sources of Single Bit Upsets (SBUs), whereas radiations can lead to SBUs and Multiple Bit Upsets (MBUs) [6]. While SBUs are composed by several Single Event Upsets (SEUs), which affect several bits in the same word, MBUs, also called burst faults, are induced by a single SEU, which affects several adjacent bits of the same data word.

In this context, fault tolerant techniques are commonly applied on the NoC to correct/mitigate permanent faults [7]. These techniques are usually based on i) mitigation through routing algorithms [8], ii) hardware reconfiguration through spare resources or default backup path [9], iii) correction through circuit replication [3] and iv) information redundancy [10]. Although the aforementioned approaches are efficient for single permanent fault, they are less adequate for multiple permanent faults. They induce high hardware costs while their mitigation capabilities are limited, as discussed in Section II. A run-time approach, named Bit-Shuffling (BiSu), reduces the impact of multiple faults, by shuffling the bits inside a flit and transferring the impact of faults on the Least Significant Bits (LSBs) [11]. Although BiSu provides a fine-grained protection, leading to high data protection, it requires a large number of additional hardware blocks in the NoC routers.

To reduce the hardware costs, while providing data protection, we propose a Region-based Bit-Shuffling technique (R-BiSu). Regarding environment conditions and the application error tolerance, we claim that the fault tolerance efficiency can be maintained with a coarse-grained mitigation. To achieve the R-BiSu technique, the NoC is divided into regions, and the shuffling/deshuffling technique is then applied at their borders. We propose a hierarchical method to compute the bit shuffling of region $X \times X$ from the bit shuffling of region $(X-1)\times(X-1)$. We design a hardware block that computes the register shuffling values, instead of using the dedicated core Intellectual Property (IP) of the routers. Last, but not least, we study in details the trade-off between the provided protection level and the required hardware costs of R-BiSu approach.

The rest of the paper is organized as follows: Section II presents the related work. Section III describes the R-BiSu method and Section IV presents the efficiency and the hardware costs of the method, when faults are injected in the NoC. Finally, Section V concludes this work.

II. Related work

The correction/mitigation of permanent faults in the NoC can be grouped into four sub-categories, summarised in the next paragraphs. Then, we present a fifth category that uses approximate approaches to mitigate faults in NoCs.

Routing algorithms are used to avoid faulty paths or faulty regions in NoCs, keeping only the healthy resources [8]. Typical techniques are adaptive routing algorithms [12] including rules to avoid congestion and deadlock during packet transmissions. Therefore, the hardware cost drastically increases with

the size of the NoCs. Using routing algorithms can be efficient, as long as the number of faults is limited. Otherwise, the latency may become higher than the acceptable limit, leading potentially to unreachable IPs. Therefore, this solution is less suitable for large NoCs and multiple faults.

Reconfiguration replaces a faulty element of the NoC by spare resources at different levels [7]. As spare resources can be used only once, these techniques have large overhead in terms of area and power consumption, while they can tolerate few faults. Other reconfiguration approaches use default-backup paths to avoid data corruptions and packet retransmissions [9] with low area and power consumption. However, the latency drastically increases under multiple faults, due to the routing complexity and, in the worst case, several IPs become unreachable. Last, NoCs can be reconfigured in degraded mode, using only the remaining healthy resources [12]. Although this method handles permanent faults with lower hardware costs, the latency, throughput and network congestion increase and some IPs can become unreachable.

Circuit replication, called N-Modular Redundancy (N-MR), replicates N times, fully or partially, the architecture and votes the replicated outputs. The most popular approach is Triple Modular Redundancy (TMR) [3], where a module is replicated three times. Despite several approaches which focus on reducing the hardware costs [13], the area and power consumption overhead remains significantly high, e.g., more than three times for TMR. Moreover, multiple faults are masked if they occur in the same module and if more than one module is affected, the voter cannot correct the output.

Information redundancy inserts additional bits inside messages using Error-Correcting Codes (ECCs). The most commonly used coding scheme for NoC is the extended Hamming code, which can detect two faulty bits and correct only one. Despite the increase of the bus size of the complete NoC, Hamming code is efficient for correcting single faults [10]. The number of correctable faulty bits can be increased by encoding the message on two dimensions [14] and interleaving several ECCs [15]. However, using ECCs to correct more than one bit dramatically increases the hardware costs [16]. As a result, the use of ECC approaches is limited against multiple faults.

The fifth category uses approximate computing for fault mitigation for NoC. In [17], an approach statically changes the assignment of lines in datapath busses, by placing the Most Significant Bits (MSBs) on the borders of the bus, to attenuate the electromagnetic influences between neighbored lines. Contrary to our method, the assignment of lines cannot be modified at run-time, and thus, external effects and manufacturing defects cannot be addressed by this technique. In [18], a multiplane NoC is proposed to ensure accurate and approximate communications using a second bufferless NoC. This method needs two parallel networks to operate increasing drastically the hardware costs. In [11], a bit-shuffling technique exploits the position of permanent faults and changes the order of bits inside a flit. However, this approach requires a large number of additional hardware blocks in the NoC routers. To reduce the hardware overhead, we propose a region-based bit-shuffling technique.

III. Proposed approach

This section presents the proposed Region-based Bit-Shuffling approach (R-BiSu) aiming at hardware costs savings. Section III-A presents the target domain and assumptions. Section III-B provides the required background on the basic BiSu technique, whereas details are available in [11]. Last, the proposed R-BiSu method is described in Section III-C.

A. Target Domain and Assumptions

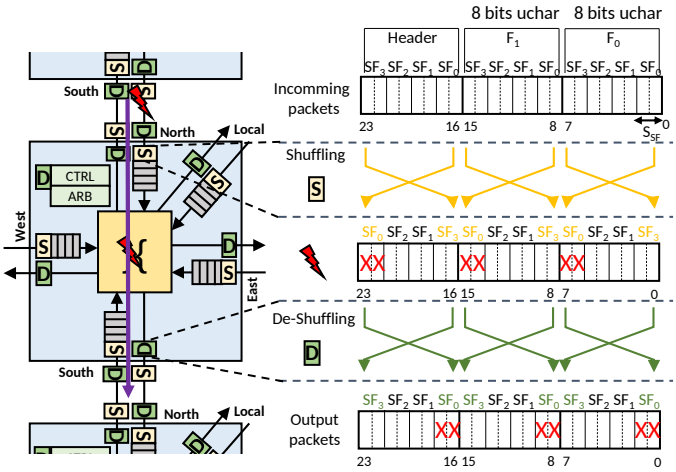
The BiSu approach mitigates multiple permanent faults, which can occur in NoC, and especially on the datapath part of the interconnect. As illustrated by the red flashes in Fig. 1, faults can be located in i) the interconnections between routers, or ii) the buffers and the crossbar within each router. Due to nanoscale technologies and power scaling, devices and components become more susceptible to multiple permanent faults [6]. As the buffers and the crossbar are the biggest components of a router [19], they have higher probability of accumulating faults due to radiation effects, manufacturing defects or other intrinsic failures. For the same reasons, interconnections are often impacted by permanent faults, usually expressed as stuck-at, short or bridge faults [20]. Such faults are managed similarly by the proposed method.

In this work, we assume that the positions of the faults (described by an error mask) are provided by methods such as Built-In Self-Test (BIST) techniques [21], which diagnose faults in interconnections and routers using Test Pattern Generator (TPG) and Output Response Analyzer (ORA) blocks. In these techniques, TPGs send test packets through the NoC, while ORAs analyze the received packets to identify if faults occurred between these two blocks, providing the fault position and type. Further details can be found in [22]. Other techniques available in the literature can be used to diagnose the permanent faults in a NoC [23]. Some methods propose to locate faults inside a NoC region [24], but their error coverage is lower than the methods able to locate faults in a single interconnection or router. As the aforementioned techniques are largely studied in the literature, they are not detailed further.

As the objective of the proposed approach is to reduce the impact of multiple faults, instead of correcting them, the targeted domains are error resilient applications used in approximate computing and communication fields, i.e., applications which can tolerate errors until a certain level, such as image processing and machine learning [25].

B. Background: The basic BiSu Technique (BiSu)

The basic BiSu technique is implemented by extending a NoC router with extra hardware blocks, i.e., Shuffler (S) and De-shuffler (D) blocks, as shown in Fig. 1-(a). The flits of size S_F bits, which transit the NoC, are divided into N_{SF} blocks of bits, called Subflit (SF). Each SF consists of S_{SF} bits. The shuffler block re-organizes the SFs with the objective of minimizing the impact of the faults. The de-shuffler block



a) Original NoC extended with the BiSu method.

b) Illustration of the BiSu method. (S_F = 8, S_{SF} = 2)

Fig. 1: NoC extended with the BiSu method.

brings back the initial order of the SFs, as illustrated in Fig. 1-(b). The BiSu technique is applied to mitigate errors on the interconnection bus and inside the router. One extra D block is added in the routing controller (CTRL) to read the routing information of the shuffled header and forward the current packet towards the expected output.

The hardware architecture of the S and D blocks is composed of N_{SF} multiplexers of N_{SF} inputs of S_{SF} bits to one output of S_{SF} bits and registers, whose values give the multiplexers selections. The S and D blocks differ in the values of their registers, which are computed with the help of an algorithm that minimizes the impact of faults. The algorithm is executed on the dedicated IP core of the router based on the error mask, i.e., the position of the permanent faults.

Faults cannot be tolerated in the header flits, since they contain control information. When the header contains several unused bits, the BiSu technique uses them to mitigate faults. When the header contains few unused bits, the BiSu technique distributes the header information into two flits. In this way, the number of unused bits is artificially increased (by duplicating the header flit) with a small impact on the NoC latency, i.e., adding a single flit in a packet. The same technique is used for sensible data, e.g., instructions. Notice that, today's NoC are typically based on large buses. Hence, the header duplication is a solution that is applied only when BiSu technique cannot provide full protection using the unused header bits.

The BiSu technique takes into account differences between both data and flit sizes, when organizing the flits inside the Network Interfaces (NIs). To achieve this, a merger block and a de-merger block are added to the NIs, and more precisely, to the packetization and de-packetization blocks always included in classic NoC. These blocks sort the data in the flits at the subflit scale, to reduce as much as possible the fault impact on the data forwarded in the flits. Further details exist in [11].

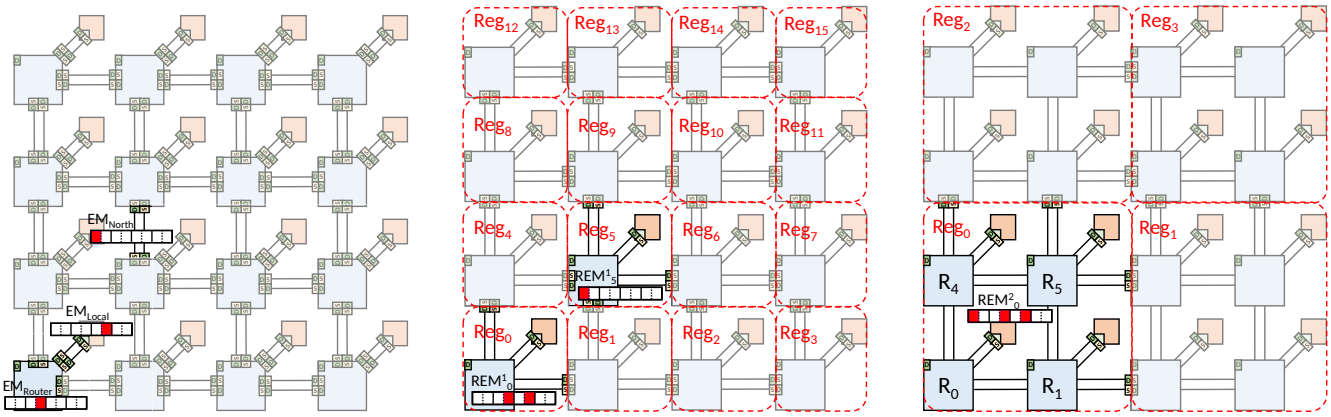
C. Region-based Bit-Shuffling approach (R-BiSu)

In this work, we propose a region-based bit-shuffling technique that reduces hardware costs in terms of area and power consumption by relaxing the mitigation efficiency of the BiSu technique. The basic idea of R-BiSu is to split the NoC into regions of routers, which are globally protected with the BiSu method. We consider regular square regions, but the method is applicable for other region sizes and shapes.

Fig. 2 illustrates the notion of R-BiSu regions for a 4×4 NoC, where the regions are distinguished by the red dotted squares. The basic BiSu technique protects each interconnection and each router, as displayed in Fig. 2a, and corresponds to a region of size 0. Fig. 2b depicts a R-BiSu configuration of size 1, i.e., the NoC is divided in regions of 1×1 including the local IP. Fig. 2c depicts a R-BiSu configuration of size 2, i.e., the NoC is divided in regions of 2×2 routers including local IP. Comparing basic BiSu and R-BiSu techniques, the basic BiSu technique requires 304 S and D blocks, whereas the 1×1 configuration reduces the number of S and D blocks to 164, reducing area and power consumption. When the size of the regions is increased to 2×2, the number of S and D blocks decreases to 80, as shown in Fig. 2c.

To implement the R-BiSu technique, the registers of the S and D blocks need to be computed taking into account all faults that are present inside a region, which can be accumulated for a packet crossing the region. To achieve that, we propose a method to compute the error mask for a region, based on the error masks of each router and interconnection. Then, we design a hardware block to compute the values of the S and D registers, reducing the pressure on the dedicated core IP of the routers, which is used for the register computations in the basic BiSu method.

1) Region Error Mask (REM) Computation: The R-BiSu method uses information regarding the faulty state of the region, given by the REM, in order to reduce as much as possible the impact of faults. This is achieved through a hierarchical method, which computes the error mask of $X \times X$ region based on error masks of $(X-1) \times (X-1)$ regions. Once the final REM is obtained, the registers of S and D blocks can be computed with low hardware costs, as shown in Section IV Fig. 2 shows an example of 4×4 NoC with 8-bit flit size and 2-bit subflit size. The NoC is affected by three faults on i) the bit number 4 of the router R_0, ii) the bit number 2 of the router R_0 local interconnection and iii) the bit number 7 of the router R_5 north interconnection. The associated error masks indicating faults (highlighted by red color) for the different region sizes are displayed on the Fig. 2. Fig. 3 depicts how the REMs are computed for size 1 and size 2 based on the available error masks of the size 0. First, as depicted by the blue squares, the REMs of size 1 (REM^1) are computed based on an OR operation between the error mask of the router and the error masks of the local, north and east interconnections. As Fig. 2b depicts, the resulted error masks indicate the faulty bits of all the 1×1 regions. For example, the REM_0^1 indicates that the bit number 2 and 4 are faulty, which correspond to



(a) Size 0 (b) Size 1×1 (c) Size 2×2

Fig. 2: Illustration of the R-BiSu technique for different region sizes.

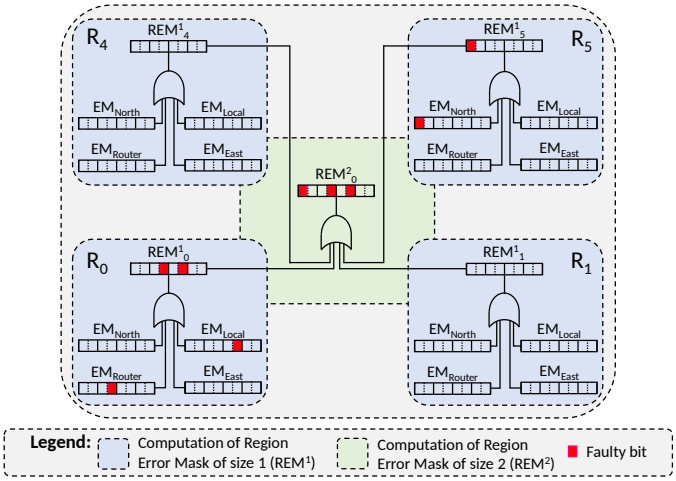


Fig. 3: Illustration of the region error mask computation.

the error masks of size 0. To compute the REMs of size 2 (REM^2), the same operation is performed, as depicted by the green square of Fig. 3, based on the REMs of size 1 (REM^1). As displayed in Fig. 2c, the obtained error mask indicates the faulty bits for all the considered 1×1 regions.

Comparing the error masks of Fig. 2, we note that the faults are individually addressed when the region size is low, guaranteeing a high efficiency of the R-BiSu method. In this case, the least significant subflit is able to tackle several faults since. In our example, only one subflit is faulty at the same time during one shuffling operation, since the faults are scattered throughout the region, in several routers and interconnections. In this case, the impact of the faults on the data is reduced as possible. However, when the region size is increased, the scattered faults are now managed by a single shuffling operation, impacting the efficiency of the R-BiSu since the faults are impacting several subflits.

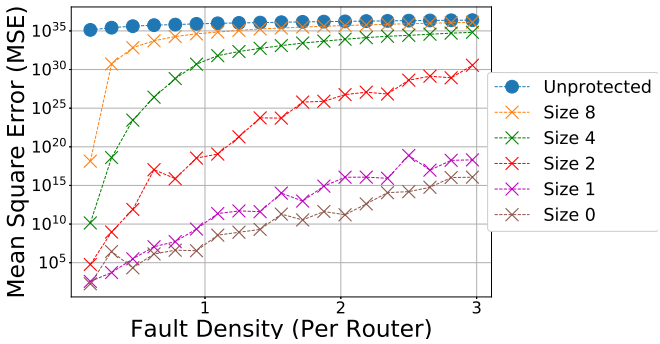
2) Computation of shuffler and de-shuffler registers: The computation of the shuffler and de-shuffler registers of the basic BiSu method is performed by the dedicated core of the IPs. The results are distributed to the shuffler and de-shuffler blocks through dedicated wires. Although the computation by the dedicated core of the IPs enables the hardware costs reduction, it increases the workload on the dedicated cores since the computation of the registers is added to the standard application workload. In order to remedy this issue, we design a dedicated hardware block inside the region that computes the value of the registers which control the multiplexers of the shuffler and de-shuffler blocks.

IV. Experimental Results

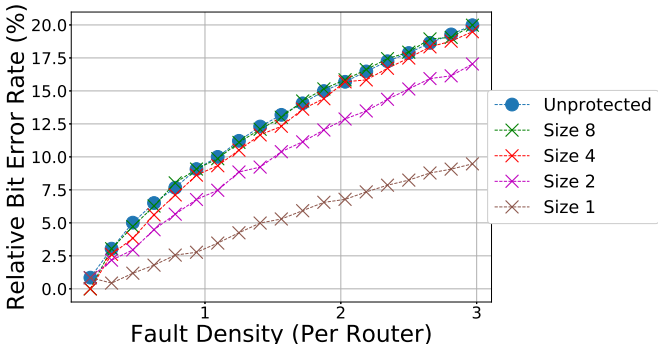
This section presents the results in terms of efficiency and hardware costs. First, we present the experimental setup. Then, we analyse the impact of the region size on the efficiency of the R-BiSu technique, when data travel on a faulty NoC, using the Mean Square Error (MSE) and Bit Error Rate (BER) metrics. Last, we present the hardware costs in terms of area and power consumption. R-BiSu has similar impact on the latency as basic BiSU, which is negligible as shown in [11].

A. Experimental Setup

For our experiments, we consider a 8×8 NoC using the XY routing algorithm where the R-BiSu method is implemented into square regions. Packets of 16 flits are injected according to the TORNADO injection model, where each IP sends a packet at any other IP. The faults are randomly injected in the NoC datapath and modeled using the stuck-at fault model [20]. We consider that the injected faults have always an impact on the data by applying a bit-flip on the affected bits. In this way, the masking effect due to data values is avoided and the fault impact is always visible. The MSE and the BER metrics, used to quantify the efficiency of the R-BiSu method, are computed based on $10,000$ fault injection sets. The CONNECT router [26] is used as baseline for the hardware implementation. The header duplication is also implemented in R-BiSu technique in order to compute the worst case in terms of hardware costs. The results are synthesized on 28 nm FDSOI technology through High Level Synthesis (HLS) tools of Mentor Graphic, targeting a clock

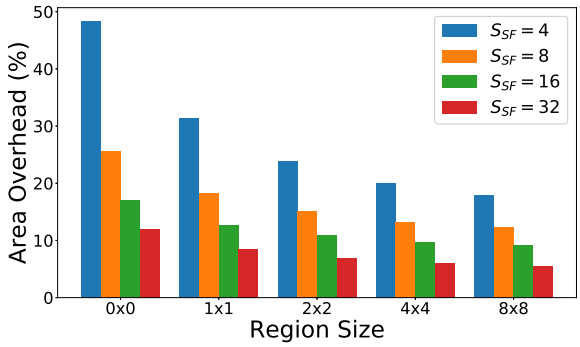


(a) Mean square error study

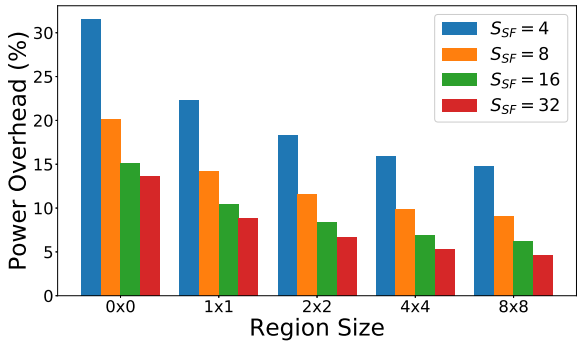


(b) Bit error rate study

Fig. 4: Experimental efficiency of the R-BiSu method at the NoC-scale. ($S_F = 64, S_{SF} = 4$)



(a) Area overhead



(b) Power overhead

Fig. 5: Hardware overhead comparison for the R-BiSu method.

frequency of 1 GHz. All simulations are performed on the Fedora 28 linux distribution with 8-cores Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz.

B. Efficiency Results

Fig. 4 depicts the MSE and the BER according to the fault density, i.e., the number of faults per router. Note that one fault per router does not mean that each router is affected by one fault, but that the mean of faults in routers is equal to one. The figures show the results obtained for different region sizes considering 64-bit flits divided into 4-bit subflits.

Fig. 4a depicts the MSE according to the fault density in the NoC. This figure shows that the size of the region has an impact on the efficiency of the R-BiSu method. It is observed that the MSE increases with the region size, until it reaches the same results with the unprotected NoC, when the density of faults is high. However, we observe that the size of the region can be increased from 0 to 1 with only a small impact on the MSE. For example, for a fault density equal to one fault per router, increasing the region size from the size 0 to the size 1 it increases the MSE only from 2.17×10^8 to 3.99×10^9.

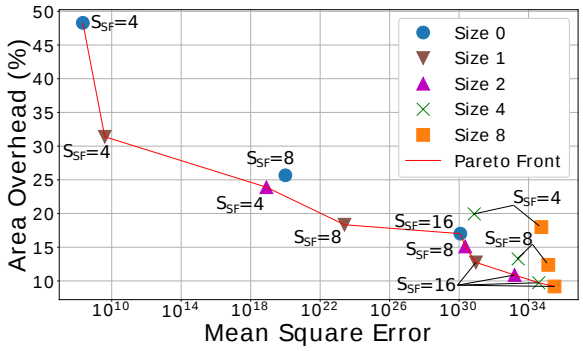
Fig. 4b displays the relative BER, taking as bases the BER obtained with the size 0, according to the fault density in the NoC. In this figure, we can note that the BER increases with the region size until reaching the same results than the unprotected case for the large region size with high fault density. Note that obtaining the same results than the unprotected case does not mean that the method is inefficient since the faults are deferred on the LSBs. For example, for a fault density equal to one fault per router, we observe that the relative BER is increased by 3% when the region size are increased from size 0 to size 1.

This metric highlights the fact that faults are mitigated through the same LSB, when several shuffling operations are successively applied on the flit. As shown in Fig 4b, this effect is more perceptible with a high fault density and with low region sizes since the number of shuffling operations is higher for one packet transmission. As the fault overlay reduces the BER, we can conclude that the method efficiency is higher when the region size is reduced.

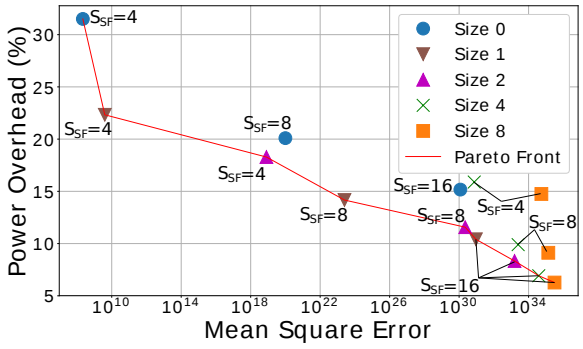
C. Hardware Results

Fig. 5 presents the hardware costs of the R-BiSu method in terms of area and power overhead for different region sizes and subflit sizes. We observe that both area and power overheads decrease with the subflit size and the region size increase. Especially when the region size increases from size 0 to size 1, the area overhead is reduced from 48.3% to 31.4% and the power overhead is reduced from 31.5% to 22.3%.

The hardware block for the value computation of the shuffling and de-shuffling registers has a small impact on the global



(a) Area Pareto front



(b) Power Pareto front

Fig. 6: Highlighting of the Pareto front at the NoC-scale.

hardware cost of the method. For example, the area and power overheads are increased by 5.3% and 5.9%, respectively, for a region of size 1, and by 1.3% and 1.4%, for a region of size 2. The latency to compute the register values depends on the number of subflits present inside a flit. For example, if the flit is composed of 16 subflits, then the latency to update the registers is equal to 370ns. When the flit is composed of 8 and 4 subflits, the latency is 120 ns and 44 ns, respectively. Therefore, these hardware blocks have low hardware costs and can be used to relax the computation pressure on the dedicated core of the IPs. Moreover, these blocks can be shared between several regions to further reduce the hardware costs.

D. Trade-off between costs and efficiency

As previously mentioned, the hardware costs reduce when the size of the region is increased. Moreover, as shown in Section IV-B, the efficiency of the method is decreased when the region size increases. Based on these observations, a Pareto front exists between the hardware costs in terms of area and power consumption and the fault mitigation efficiency of the proposed method. Fig. 6 plots the Pareto front for different subflit sizes and region sizes considering 64-bit flits and a fault density equal to one fault per router. From the obtained result, we observe that there are cases where the basic BiSu does not belong to the Pareto front. Furthermore, the region size can be increased from size 0 to size 2 with a small impact on the efficiency, while the area and the power overheads are reduced from 48% to 33% and from 34% to 22%, respectively. Moreover, an increase of the subflit size, reducing hardware overheads, has a higher impact on the efficiency. Last, we conclude that increasing the region size is a better way to reduce the hardware overheads than increasing the subflit size.

V. Conclusion

In this work, we proposed a region-based bit-shuffling method with reduced hardware costs. To achieve that, a hierarchical method is proposed to compute the mask error of a complete region. In addition, we design a hardware block to compute the values of the registers of shuffler and de-shuffler blocks, relaxing the pressure on the dedicated core of the IPs. The obtained results show that an increase of the region size from size 0 to size 2 can reduce the hardware costs in terms of area and power consumption from 48% to 33% and from 34% to 22%, respectively, with a small impact on the MSE.

References

[1] J. Xu, W. Wolf, J. Henkel, and S. Chakradhar, "A Methodology for Design, Modeling, and Analysis of Networks-on-chip," in *IEEE Int. Symp. Circuits and Syst. (ISCAS)*, vol. 2, pp. 1778–1781, May 2005.

[2] M. T. Bohr, "Logic Technology Scaling to Continue Moore's Law," in *IEEE Electron Devices Technol. and Manuf. Conf. (EDTM)*, pp. 1–3, Mar. 2018.

[3] E. Dubrova, *Fault-Tolerant Design*. Springer-Verlag New York, 2013.

[4] S. Kundu and S. Chattopadhyay, *Network-on-Chip: The Next Generation of System-on-Chip Integration*. Taylor & Francis, 2014.

[5] "Space Product Assurance: Techniques for Radiation Effects Mitigation in AASIC and FPGAs Handbook," tech. rep., ESA Requirements and Standards Division, Sept. 2016.

[6] "Single Event Effects Mitigation Techniques Report," *Federal Aviation Admin., William J. Hughes Tech. Center*, Feb. 2016.

[7] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Self-Healing Hardware Systems: A Review," *Microelectron. J.*, vol. 93, p. 104620, 2019.

[8] B. Fu, Y. Han, H. Li, and X. Li, "ZoneDefense: A Fault-Tolerant Routing for 2-D Meshes Without Virtual Channels," *IEEE Trans. on Very Large Scale Integration (VLSI) Syst.*, vol. 22, pp. 113–126, Jan. 2014.

[9] M. Ebrahimi, M. Daneshtalab, J. Plosila, and H. Tenhunen, "Minimal-Path Fault-Tolerant Approach Using Connection-Retaining Structure in Networks-on-Chip," in *IEEE/ACM Int. Symp. on Networks-on-Chip (NOCS)*, pp. 1–8, Apr. 2013.

[10] G. Liva, L. Gaudio, T. Ninacs, and T. Jerkovits, "Code Design for Short Blocks: A Survey," *Computing Research Repository (CoRR)*, Oct. 2016.

[11] R. Mercier, C. Killian, A. Kritikakou, Y. Helen, and D. Chillet, "Multiple Permanent Faults Mitigation Through Bit-Shuffling for Network-on-Chip Architecture," in *IEEE Int. Conf. on Comput. Design (ICCD)*, pp. 205–212, Oct. 2020.

[12] Z. Chen, Y. Zhang, Z. Peng, and J. Jiang, "A Deterministic-Path Routing Algorithm for Tolerating Many Faults on Wafer-Level NoC," in *Des. Automat. Test in Europe Conf. Exhib. (DATE)*, pp. 1337–1342, Mar. 2019.

[13] A. Mukherjee and A. S. Dhar, "Triple Transistor Based Triple Modular Redundancy With Embedded Voter Circuit," *Microelectron. J.*, vol. 87, pp. 101 – 109, May 2019.

[14] X. Chen, Z. Lu, Y. Lei, Y. Wang, and S. Chen, "Multi-Bit Transient Fault Control For NoC Links Using 2D Fault Coding Method," in *IEEE/ACM Int. Symp. on Networks-on-Chip (NOCS)*, pp. 1–8, Aug. 2016.

[15] Q. Yu and P. Ampadu, "Adaptive Error Control for NoC Switch-to-Switch Links in a Variable Noise Environment," in *IEEE Int. Symp. on Defect and Fault Tolerance of VLSI Syst.*, pp. 352–360, 2008.

[16] A. Sánchez-Macián, P. Reviriego, and J. A. Maestro, "Hamming SEC-DAED and Extended Hamming SEC-DED-TAED Codes Through Selective Shortening and Bit Placement," *IEEE Trans. on Device and Mater. Rel*, vol. 14, pp. 574–576, Mar. 2014.

[17] A. Najafi, L. Bamberg, A. Najafi, and A. Garcia-Ortiz, "Integer-Value Encoding for Approximate On-Chip Communication," *IEEE Access*, vol. 7, pp. 179220–179234, Dec. 2019.

[18] L. Wang, Y. Wang, and X. Wang, "An Approximate Multiplane Network-on-Chip," in *Des. Automat. Test in Europe Conf. Exhib. (DATE)*, pp. 234–239, Mar. 2020.

[19] A. DeOrio, D. Fick, V. Bertacco, D. Sylvester, D. Blaauw, J. Hu, and G. Chen, "A Reliable Routing Architecture and Algorithm for NoCs," *IEEE Trans. on Comput.-Aided Des. of Integr. Circuits and Syst. (TCAD)*, vol. 31, pp. 726–739, May 2012.

[20] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Elsevier, 2004.

[21] H. J. Mohammed, W. N. Flayyih, and F. Z. Rokhani, "Tolerating Permanent Faults in the Input Port of the Network on Chip Router," *J. of Low Power Electron. and Appl.*, vol. 9, pp. 1–11, Feb. 2019.

[22] B. Bhowmik, S. Biswas, J. K. Deka, and B. B. Bhattacharya, "A Low-Cost Test Solution for Reliable Communication in Networks-on-Chip," *J. of Electron. Testing*, vol. 35, pp. 215–243, Apr. 2019.

[23] D. Xiang, K. Chakrabarty, and H. Fujiwara, "A Unified Test and Fault-Tolerant Multicast Solution For Network-on-Chip Designs," in *IEEE Int. Test Conf. (ITC)*, pp. 1–9, Nov. 2016.

[24] M. B. Hervé, M. Moraes, P. Almeida, M. Lubaszewski, F. L. Kastensmidt, and É. Cota, "Functional Test of Mesh-Based NoCs with Deterministic Routing: Integrating the Test of Interconnects and Routers," *J. of Electron. Testing*, vol. 27, no. 5, pp. 635–646, 2011.

[25] A. B. Ahmed, D. Fujiki, H. Matsutani, M. Koibuchi, and H. Amano, "AxNoC: Low-power Approximate Network-on-chips Using Critical-path Isolation," in *IEEE/ACM Int. Symp. on Networks-on-Chip (NOCS)*, no. 6, pp. 1–8, Oct. 2018.

[26] M. K. Papamichael and J. C. Hoe, "CONNECT: Re-examining Conventional Wisdom for Designing Nocs in the Context of FPGAs," in *ACM/SIGDA Int. Symp. Field Program. Gate Arrays (FPGA)*, pp. 37–46, Feb. 2012.