HAL
open science

A Region-Based Bit-Shuffling Approach Trading Hardware Cost and Fault Mitigation Efficiency

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

To cite this version:

Romain Mercier, Cédric Killian, Angeliki Kritikakou, Youri Helen, Daniel Chillet. A Region-Based Bit-Shuffling Approach Trading Hardware Cost and Fault Mitigation Efficiency. DFT 2021 - 34th IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, Oct 2021, athens, Greece. pp.1-4, 10.1109/DFT52944.2021.9568366 . hal-03500395

HAL Id: hal-03500395

https://hal.science/hal-03500395v1

Submitted on 22 Dec 2021

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.

A Region-Based Bit-Shuffling Approach Trading Hardware Cost and Fault Mitigation Efficiency

† 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. Indeed, 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, 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. The obtained results show that the area and power overheads can be reduced from 48% to 33% and from 34% to 22%, respectively, with a small impact on the Mean Square Error (MSE).**

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

I. Introduction

In the current technology era, Network-on-Chips (NoCs) became more sensitive to permanent faults. The engraving thinness causes more and more hardware defects, due to manufacturing process [1]. During system operation, aging defects [2] and radiations [3] become additional sources of permanent faults. Manufacturing and aging defects are well known sources of Single Bit Upsets (SBUs), i.e. several bits affected by several Single Event Upsets (SEUs), whereas radiations can lead to SBUs and Multiple Bit Upsets (MBUs) [4], i.e. several bits affected by a single SEU.

In this context, fault tolerant techniques are commonly applied on the NoC to correct/mitigate permanent faults. Routing algorithms [5] are used to avoid faulty paths increasing drastically the hardware costs with the size of the NoCs. Then, this solution is less suitable for large NoCs and multiple faults which can lead to high latency and unreachable Intellectual Property (IP). Reconfiguration techniques replace faulty elements by spare resources [6] which increase drastically the hardware cost while few faults can be tolerated. Default-backup path [7] can be used to limit the hardware costs but it impacts the performances of the NoC and potentially leads to unreachable IP. Circuit replication [1], called N-Modular Redundancy (N-MR), is used to mask faults by replicating N times the architecture and votes the replicated outputs. This solution induces significant hardware costs to mask only faults which occur in a same module. Error-Correcting Codes (ECCs) are used to detect and correct faults by inserting additional bits in the messages increasing dramatically the hardware costs. The most commonly used coding scheme for NoC is the extended Hamming code, which can detect two faulty bits and correct only one [8]. Approximate computing can be used for fault mitigation in NoCs [9]. In [10], a bit-shuffling technique, named Bit-Shuffling (BiSu), exploits the position of permanent faults at run-time 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 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/de-shuffling 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 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.

II. Proposed approach

This section presents the proposed Region-based Bit-Shuffling approach (R-BiSu) aiming at hardware costs savings. Section II-A presents the target domain and assumptions. Section II-B provides the required background on the basic BiSu technique, whereas details are available in [10]. Last, the proposed R-BiSu method is described in Section II-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 in ii) the buffers and the crossbar within each router. 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 [11]. As the objective

978-1-6654-1609-2/21/$31.00 ©2021 IEEE

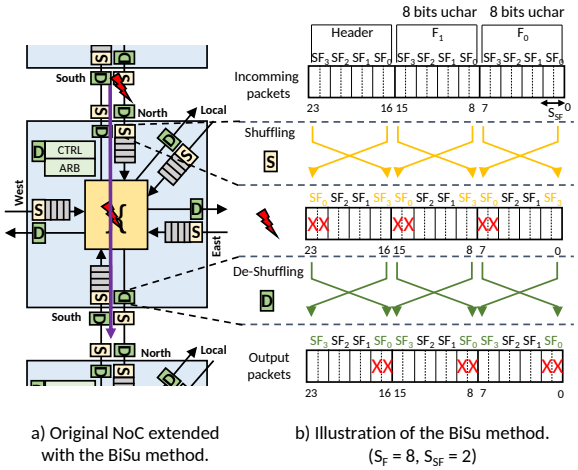


Fig. 1: NoC extended with the BiSu method.

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 such as image processing and machine learning [12].

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. 1a. 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 brings back the initial order of the SFs, as illustrated in Fig. 1b. 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. Further details are given in [13] [10].

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.

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, i.e. size 0, as displayed in Fig. 2a. Fig. 2b and 2c depict respectively a R-BiSu configuration of size 1 and 2, i.e., the NoC is respectively divided in regions of 1×1 and 2×2 including the local IPs. Comparing basic BiSu and R-BiSu techniques, the basic BiSu technique requires 304 S and D blocks, whereas the 1×1 and 2×2 configurations reduce respectively the number of S and D blocks to 164 and 80, reducing area and power consumption.

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 III. Fig. 2 shows an example of 4×4 NoC with 8-bit flit size and 2-bit subflit size where the NoC is affected by three faults. The associated error masks are displayed indicating faults (highlighted by red color) for the different region sizes.

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. 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). Comparing the error masks of Fig. 2, we note that the faults are individually addressed by several shuffling operations when the region size is low, guaranteeing a high efficiency of the R-BiSu method. However, when the region size is increased, the scattered faults are managed by a single shuffling operation, impacting the efficiency of the R-BiSu since the faults are impacting several subflits.

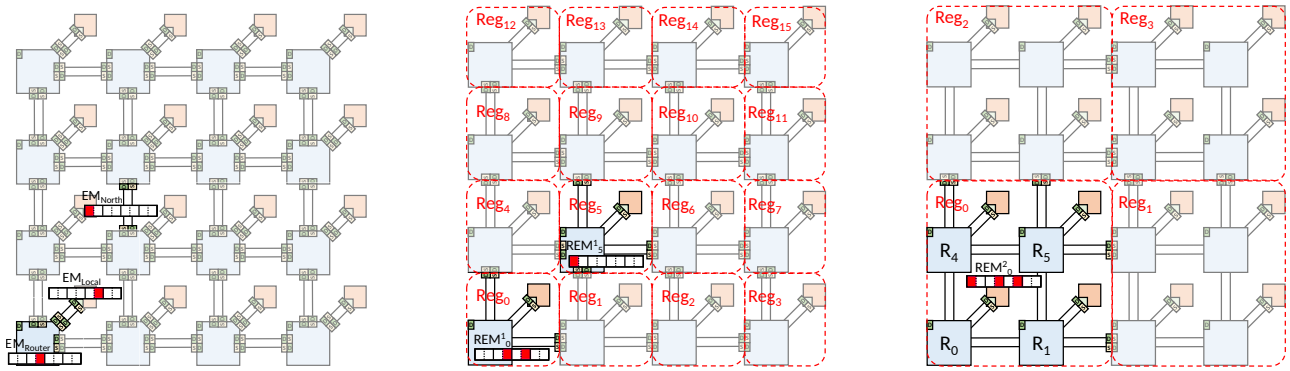
2) Computation of shuffler and de-shuffler registers: The computation of the shuffler and de-shuffler registers of the basic BiSu method is performed by the dedicated core of the IPs. The results are distributed to the shuffler and de-shuffler blocks through dedicated wires. Although the computation 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.

III. Experimental Results

This section presents the results in terms of efficiency and hardware costs. First, we present the experimental setup. Then, we analyze 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 lower impact on the latency than basic BiSu, which is negligible as shown in [13] [10].

A. Experimental Setup

For our experiments, we consider a 8×8 NoC using the XY routing algorithm where the R-BiSu method is implemented



(a) Size 0 (b) Size 1×1 (c) Size 2×2

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

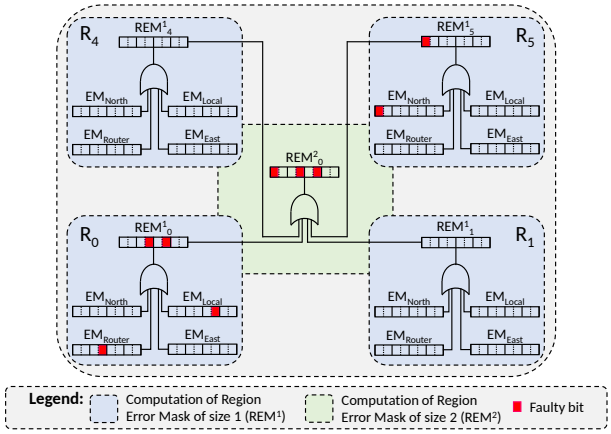


Fig. 3: Illustration of the region error mask computation.

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 [14]. 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 [15] 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 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

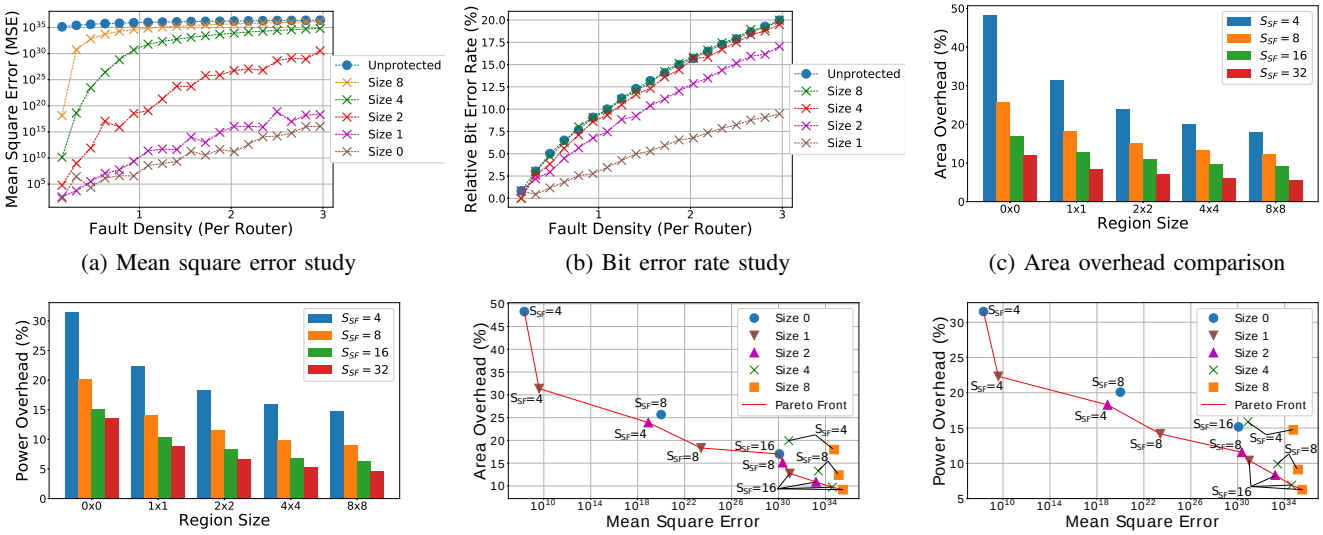
Fig. 4a and 4b respectively depict the MSE and the BER according to the fault density for different region sizes considering 64-bit flits divided into 4-bit subflits.

Fig. 4a 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.

Fig. 4b displays the relative BER considering the BER with size 0 as reference. 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 Least Significant Bits (LSBs). 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. 4c and 4c present respectively the area and power overhead of the R-BiSu method 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 hardware block for the value computation of the shuffling and de-shuffling registers has a small impact on the global 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



(a) Mean square error study

(b) Bit error rate study

(c) Area overhead comparison

(d) Power overhead comparison

(e) Area Pareto front highlighting

(f) Power Pareto front highlighting

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

core of the IPs. Moreover, these blocks can be shared between several regions to further reduce the hardware costs.

D. Costs and efficiency trade-off

Fig. 4e and 4f plot respectively the area and power 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.

IV. Conclusion

In this work, we proposed a fault mitigation based on region bit-shuffling method, with a reduction of hardware cost compare to a full mitigation. To achieve that, a hierarchical method is proposed to compute the error masks 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 with a small impact on the MSE.

Acknowledgment

This work is supported by the Directorate General of Armaments (DGA) and the ANR SHNoC project, grant ANR-18-CE25-0006 of the French Agence Nationale de la Recherche.

References

[1] E. Dubrova, *Fault-Tolerant Design*. Fault-Tolerant Design, Springer, 2013.

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

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

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

[5] 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.

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

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] R. Mercier, C. Killian, A. Kritikakou, Y. Helen, and D. Chillet, "BiSuT: A NoC-Based Bit-Shuffling Technique for Multiple Permanent Faults Mitigation," *IEEE Trans. on Comput.-Aided Des. of Integr. Circuits and Syst. (TCAD)*, pp. 1–1, 2021.

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

[15] 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.