Evaluation of Multiple Bit Upset Tolerant Codes for NoCs Buffering

Felipe Silva, Walter Magalhães, Jarbas Silveira, João Marcelo Ferreira, Philippe Magalhães
DETI-UFC Federal University of Ceará Fortaleza, Brazil
{gaspar, jarbas, joaomarcelo, philippe}@lesc.ufc.br; walter@gtel.ufc.br

Otavio A. Lima Jr
Digital Systems Laboratory-IFCE
Federal Institute of Ceará
Fortaleza, Brazil
otavio@ifce.edu.br

César Marcon
PPGCC-PUCRS
Pontifical Catholic University of Rio Grande do Sul
cesar.marcon@pucrs.br

Abstract—Newest technologies of integrated circuits manufacture allow billions of transistors arranged in a single chip enabling to implement a complex parallel system, which requires a communication architecture with high scalability and a high degree of parallelism, such as a Network-on-Chip (NoC). As the integration technology scales down, the probability of Multiple Cell Upsets (MCUs) increases. NoC buffers are exposed to MCUs induced by different sources. In this paper, we evaluate Error Correction Codes (ECCs) to protect NoC buffers from permanent MCUs. We guide our evaluation to measure the area and power overhead of each implementation of ECC, as well as their correction and detection rates. We conducted experiments varying buffer parameters (16 and 32-bits flit length) and three different protection codes. The results show that the costs of adding an ECC in each input buffer of the NoC are justified by the decrease of error occurrence for systems exposed to MCU events.

Keywords—Network on chips; Fault Tolerance; Error Correction Code

I. INTRODUCTION

The accelerated development of VLSI technologies has enabled the integration of tens or hundreds of complex logic cores like processors, memories, and specialized logic blocks in a System-on-Chip (SoC). SoCs demand a highly scalable and parallelized communication infrastructure, which is unreachable to some technologies like shared buses, hierarchical buses, and point-to-point connections. An embedded network called Network-on-Chip (NoC) has come to cope with on-chip communication problems, switching communication paradigm from physical connections to a packet-oriented network [1].

NoCs have greater flexibility, scalability and parallelism than other solutions [2], but a very simple architecture composed of only four components: links, network interfaces and routers. On-going packets are stored into routers data buffers, which occupy large chip area, to cope with the network charge fluctuations. For instance, considering a relatively small number of buffer entries per router (e.g. 16) where each entry stores 64 bytes of data, a network with 64 nodes requires 64 KB of buffer storage [3].

NoC buffers are exposed to Multiple Cell Upsets (MCUs) induced by different sources. For example, a highly charged particle striking at a memory cell [5], directed ionization and nuclear recoil after the passage of a high-energy ion [6]. As the integration technology scales down, the probability of MCUs increases [4]. Furthermore, manufacturing process variability can also affect the occurrence of MCUs [4]. One of the protection techniques used to deal with MCUs is the employment of a robust Error Correction Code (ECC), which can be improved by the addition of built-in current sensors designed to detect unexpected variations in the current of memory blocks [7][8].

In this paper, we evaluate ECCs to protect NoC buffers from permanent MCUs. We guide our evaluation to measure area and power overhead of each ECC implementation, as well as their correction and detection rates. We conducted experiments varying buffer parameters for a wide range of test scenarios. We verify one million words generated pseudo-randomly for each test scenario. Failures are positioned in adjacent cells to mimic the structure of MCUs. The experimental results enable the designers to evaluate the adequate ECC for a given NoC configuration.

We divided the related work into two important groups. The first one is composed of works using ECCs on NoCs. The second one consists of ECCs designed to correct single and multiple errors. Previous works have dealt with transient errors in NoC links by using ECCs in data link layer (hop-to-hop ECC) [9][10], as well as network layer [11]-[13], which protects a data packet by correcting errors only in the destination node. In [14], a SRAM/STT-MRAM IO buffer is proposed, and an SEC-DED code is used to deal with the errors introduced due to the stochastic switching of the STT-MRAM buffers.

The remaining of this paper is organized as follows. In Section II we present the description of the ECCs applied in the NoC. The experimental setup is presented in Section III. The discussion about the results obtained in the synthesis process and the key findings are presented in Section IV e V, respectively.

II. DESCRIPTION OF THE ERROR CORRECTION CODES APPLIED

In the few recent years, ECCs have been widely applied to fault tolerant systems due to their outstanding benefits for solving Single Cell Upset (SCU) and MCU with low implementation cost. In this paper, we evaluate three ECCs: Extended Hamming, Matrix and Column-Line-Code (CLC) to

978-1-5090-5859-4/17/$31.00 ©2017 IEEE

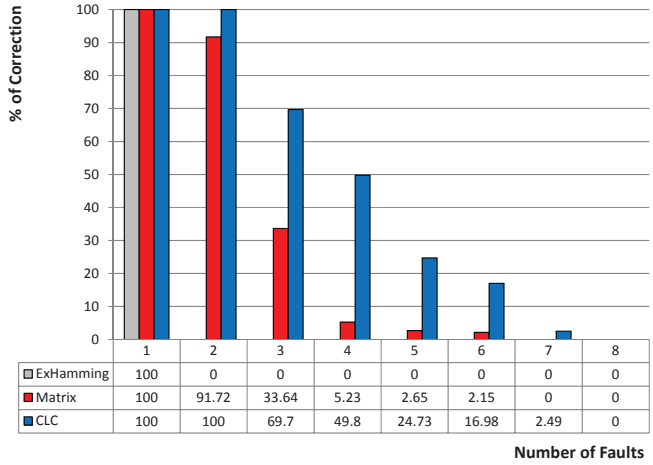
protect buffers of a NoC [15] from Multiple Bit Upsets (MBUs).

The Hamming code was one of the first codes employed to deal with failures in computer systems, being capable of correct SCU events. By adding parity, the Hamming code can be extended to improve its detection capability to detect Double Cell Upsets (DCUs) events. This version of the code is called Extended Hamming (ExHamming), which has a very simple logic and low implementation cost.

The Matrix code uses concepts of Hamming code and Parity to form a two dimensional scheme for correcting and detecting some patterns of MCU. The Matrix code applied in this paper is based on the model proposed in [16], where the primary data input is divided into groups of bits. Each group is codified by Hamming and Parity forming the matrix. Although the Matrix code presents better correction and detection performance than ExHamming code, it is more complex, having more redundancy bits and consuming more area and energy.

The last code applied was the CLC code based in [17]. This code utilizes concepts of ExHamming and Parity forming, such as Matrix code, a two-dimensional scheme. CLC code uses more redundancy bits than Hamming and Matrix, enabling to detect and correct more types of MCUs. However, the bigger number of redundancy bits in comparison with Matrix and ExHamming codes increases area consumption and power dissipation, making CLC the ECC with a bigger implementation cost of the three methods analyzed.

Fig. 1 and Fig. 2 show the correction and detection capabilities of the three codes evaluated here.

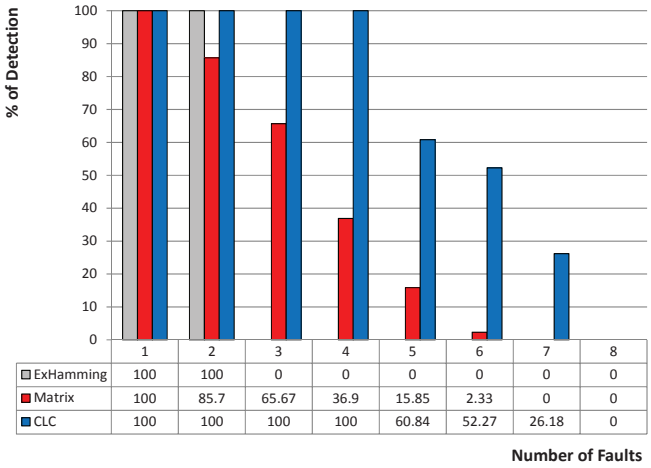


	1	2	3	4	5	6	7	8
ExHamming	100	0	0	0	0	0	0	0
Matrix	100	91.72	33.64	5.23	2.65	2.15	0	0
CLC	100	100	69.7	49.8	24.73	16.98	2.49	0

Fig. 1 Correction Analysis for ExHamming, Matrix, and CLC codes.

The results from Matrix and CLC were collected from previous experiments performed by the authors in [17]. ExHamming code is a well-known Single Error Correction and Double Error Detection code (SEC-DED).

Fig. 1 and Fig. 2 exhibits that CLC is the most robust code from all presented, being capable of correcting and detecting an average of nearly 16% and 29%, respectively. These error percentages are larger than Matrix, and about 33% and 32% bigger than ExHamming. It can be explained due mainly to the bigger number of redundancy in its codified word, which allows the detection and correction of more complex cases of MCUs.



	1	2	3	4	5	6	7	8
ExHamming	100	100	0	0	0	0	0	0
Matrix	100	85.7	65.67	36.9	15.85	2.33	0	0
CLC	100	100	100	100	60.84	52.27	26.18	0

Fig. 2 Detection Analysis for ExHamming, Matrix, and CLC codes.

III. Experimental Setup

It is well known that fault-tolerant techniques consume many circuit resources. Leem et al. [18] advocate the use of fault tolerance in critical parts of design, primarily those structures dealing with crucial systems information. In this paper, we evaluate the previously described ECCs implemented on the NoC's router [15] to help the designers to assess the cost of protecting buffers.

We analyze a NoC with 16 bits and 32 bits data flits, implemented on an 11×11 regular grid topology, where each router has five communication channels (LOCAL, EAST, WEST, NORTH, SOUTH), 16 flits buffers depth, wormhole switching, and On-off flow control. The routing algorithm is minimum and deployed by distributed tables. The encoder (Enc ECC) and decoder (Dec ECC) of the ECCs were implemented in the buffers of the NoC's router, where is more probable to happen SCU and MCU events. Fig. 3 exposes the NoC router structure.

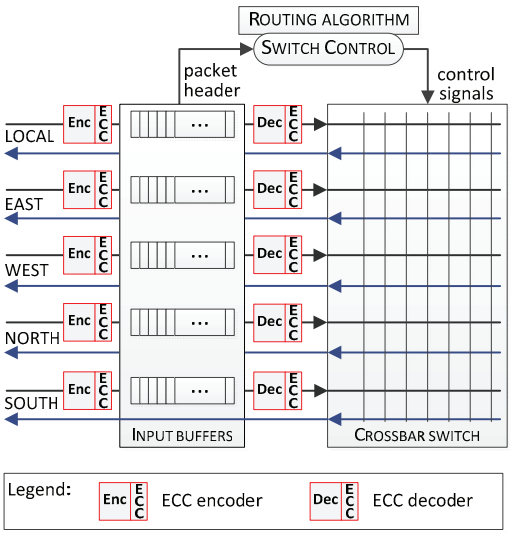


Fig. 3 NoC's router structure analyzed in the experiments.

Before being stored in the NoC's buffers, the data flits are encoded by the ECC. When the information in the buffer is ready to be processed, the encoded data is handled by the decoder, which verifies the occurrence of bit flips and in the case of detected errors, the information is restored.

IV. Experimental Results and Analysis

For cost analysis, we designed the NoC and ECC codes in VHDL and synthesized for 65 nm technology with the *Encounter RTL Compiler* from Cadence. Table I depicts the synthesis results of the NoC router and channels with ECC protection. For fault experimentation, the ECC's presented in section II were applied in a single router. The NoC buffers were replied in MATLAB, and submitted to one million words pseudo-randomly generated for each one of the eight error test scenarios. The results for error coverage achieved were similar to the illustrated in Fig 1 and Fig 2.

TABLE I. SYNTHESIS RESULTS

Design tested	Flit size (bits)	Cells	Area (um²)	Power (mW)
Full router system	16	3553	42295	2.890
	32	6694	78802	4.670
Single router buffer	16	566	6667	0.401
	32	1144	13012	0.745
Single router buffer with ExHamming	16	1212	12776	0.703
	32	1993	21575	1.283
Single router buffer with Matrix	16	1658	17473	1.043
	32	2830	30633	1.784
Single router buffer with CLC	16	2405	23895	1.256
	32	4202	42395	2.102

Table I presents the cost of a Full router system, which is formed by: the Five buffers (without an ECC), Routing algorithm, Switch control and Crossbar, and the cost of a single router buffer with each ECC applied. Table I clarifies that CLC has larger overhead results in the synthesis process, which is explained by the fact that CLC is the most complex code from all others analyzed. It requires more redundancy bits than the others to detect and correct more MCUs patterns. Matrix code proved to be an intermediate solution since the Matrix code has bigger logical structure than the ExHamming code; however, it employs fewer redundancy bits than CLC, which reduces its cost. Finally, the ExHamming has lowest cost numbers in the analysis.

Table II exposes the increase for the NoC's router brought by the implementation of an ECC in a single buffer.

TABLE II. COST INCREASE IN THE NOC ROUTER

Design tested	Flit size (bits)	Increase of cells (%)	Increase of area (%)	Increase of power (%)
Single router buffer with ExHamming	16	18.18	14.44	10.44
	32	12.68	10.86	11.52
Single router buffer with Matrix	16	30.73	25.54	22.21
	32	25.18	22.36	22.24
Single router buffer with CLC	16	51.75	40.73	29.58
	32	45.68	37.28	29.05

Table II shows that adding an ECC in a single buffer adds significant overhead in a router, adding it to all NoC's buffers is a costly design choice. Aiming to facilitate this issue, the application of an ECC could be restricted to a few router buffers, especially those who present more fault tendencies. It is up to the NoC designer to select which buffers must be protected by analyzing the systems requirements.

We verify the synthesis results by using a metric called *Total Coverage per Cost* (*TCC*), which measure the codes suitability to deal with SCUs and MCUs. The Equation 1 was applied to evaluate the TCC of each single router with an ECC (Exhamming, Matrix and CLC) dividing the error coverage of the ECC applied (product between detection rate and correction rate) by the cost of the structure (product between area and power consumed). The results achieved in Table I, jointly with the Fig. 1 and Fig. 2, were used to compose TCC, similar to the metric proposed in [17].

$$TCC = \frac{100 \times detection\_rate \times correction\_rate}{area \times power} \quad (1)$$

Fig. 4 and Fig. 5 show the results generated by Equation 1 for the data collected with 16 bits and 32 bits flit, respectively. We analyze eight error scenarios as in the correction and detection rates showed in Fig. 1 and Fig. 2.

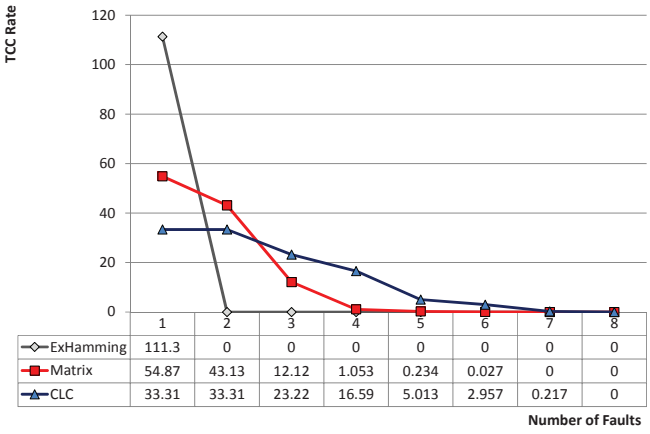


Fig. 4 TCC Rate for NoC with 16-bit flit.

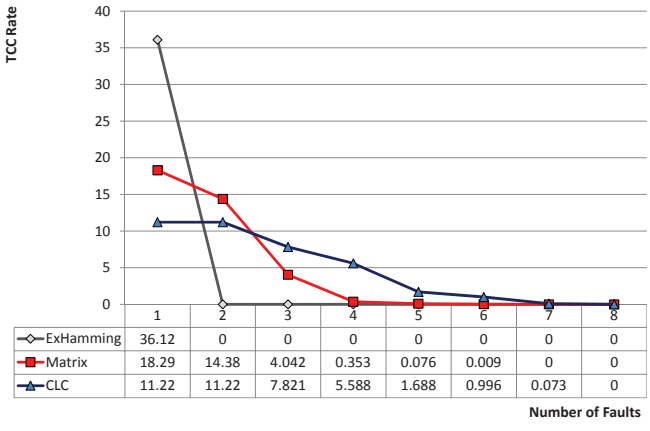


Fig. 5 TCC Rate for NoC with 32-bit flit.

As we can see in Fig. 4 and Fig. 5, ExHamming code achieved the best TCC results for one fault situation, for both flit sizes. It can be explained by the fact that ExHamming has lower implementation cost than the other codes. However, for

the other seven situations, ExHamming had TCC equals to 0 since it cannot correct MCUs. Meanwhile, Matrix had best results for two faults scenario, once Matrix is not able to correct many patterns of multiple errors making the *TCC* rate sharply falling as long as the number of faults increases.

Finally, CLC achieved lower results than other codes for the two initial scenarios, due to its high number of redundancy bits, which makes its implementation cost higher than the other codes. However, CLC was designed for critical situations, where the occurrence of MCUs is a major concern for the system reliability, mainly to avoid critical failures. From that, the *TCC* analysis for the CLC proved to be satisfactory, as from three to seven fault scenarios, CLC presented the best *TCC* rates for both flit sizes.

V. Conclusion

This paper analyzed the implementation of ECC in a NoC router to protect buffers from SCU and MCU events. The results presented in Section IV show that the implementation of an ECC in one buffer of a NoC increases its synthesis cost due to the addition of more combinational logic and redundancy bits. However, as it was exposed in Section I, the occurrence of errors in NoCs are a major concern for future application, due to their components may be exposed to SCU and MCU events. As a possible solution for this, the appliance of these techniques can be restricted to some critical areas of a NoC to minimize the increase of area and power overhead. As future work, we propose to evaluate automatic design tools to decide which paths must be protected to achieve fault tolerance and the cost requirements.

Finally, the *TCC* results presented in Fig. 4 and Fig. 5 help to choose which ECC is more suitable to a NoC vulnerable to buffers failures. ExHamming coding proved to be the more suitable option for SCUs, Matrix code for DCU and CLC presented the best overall results, since from three to seven errors their *TCC* results outperformed ExHamming and Matrix codes.

References

[1] Salaheldin, K. Abdallah, N. Gamal, H. Mostafa. **Review of NoC-based FPGAs Architectures**. *IEEE International Conference on Energy Aware Computing Systems & Applications (ICEAC)*, pp. 1-4, 2015.

[2] S. Jiang, Y. Liu, J. Luo, H. Cheng, G. Luo. **Study of Fault-Tolerant Routing Algorithm of NoC based on 2D-Mesh Topology**. *IEEE International Conference on Applied Superconductivity and Electromagnetic Devices (ASEMD)*, pp. 189-193, 2013.

[3] T. Moscibroda, O. Mutlu. **A Case for Bufferless Routing in On-chip Networks**. *Annual International Symposium on Computer Architecture (ISCA)*, pp. 196-207, 2009.

[4] D. Radaelli, H. Puchner, S. Wong, S. Daniel. **Investigation of multibit upsets in a 150 nm technology SRAM device**. *IEEE Transactions on Nuclear Science,* vol. 52, n. 6, pp. 2433-2437, Dec. 2005.

[5] J. Maestro, P. Reviriego. **Study of the Effects of MBUs on the Reliability of a 150 nm SRAM Device**. *ACM/IEEE Design Automation Conference (DAC)*, pp. 930-935, 2008.

[6] R. Hentschke, F. Marques, F. Lima, L. Carro, A. Susin, R. Reis, **Analyzing Area and Performance Penalty of Protecting Different Digital Modules with Hamming Code and Triple Modular Redundancy**. Symposium on Integrated Circuits and Systems Design (SBCCI), pp. 95-100, 2002.

[7] C.-L. Hsu, M.-H. Ho, C.-F. Lin. **Novel Built-In Current-Sensor-Based Testing Scheme for CMOS Integrated Circuits**. *IEEE Transactions on Instrumentation and Measurement*, vol. 58, n. 7, pp. 2196-2208, Jul. 2009.

[8] D. Toro, M. Arzel, F. Seguin, M. Jezequel. **Soft Error Detection and Correction Technique for Radiation Hardening Based on C-element and BICS**. *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, n. 12, pp. 952-956, Dec. 2014.

[9] L. Li, N. Vijaykrishnan, M. Kandemir, M. Jrwin. **Adaptive error protection for energy efficiency**. *International Conference on Computer Aided Design (ICCAD)*, pp. 2-7, 2003.

[10] Q. Yu, P. Ampadu. **Adaptive Error Control for Nanometer Scale Network-on-Chip Links**. *IET Computers & Digital Techniques*, vol. 3, n. 6, pp. 643-659, Apr. 2009.

[11] M. Ali, M. Welzl, S. Hessler, S. Hellebrand. **An Efficient Fault-Tolerant Mechanism to Deal with Permanent and Transient Failures in a Network-on-Chip**. *International Journal of High Performance Systems Architecture*, vol. 1, no. 2, pp. 113-123, Oct. 2007.

[12] A. Sanusi, M. Bayoumi. **Smart-Flooding: A Novel Scheme for Fault-Tolerant NoCs**. *IEEE International SOC Conference (SOCC)*, pp. 259-262, 2009.

[13] Y.-C. Lan, M. Chen, W.-D. Chen, S.-J. Chen, Y.-H. Hu. **Performance-Energy Tradeoffs in Reliable NoCs**. *International Symposium on Quality Electronic Design (ISQED)*, pp. 141-146, 2009.

[14] T. Majumder, M. Suri, V. Shekhar. **NoC Router Using STT-MRAM based Hybrid Buffers with Error Correction and Limited Flit Retransmission** *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2305-2308, 2015.

[15] J. Ferreira, J. Silveira, J. Silveira, R. Cataldo, T. Webber, F. Moraes, C. Marcon. **Efficient Traffic Balancing for NoC Routing Latency Minimization**. *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2599-2602, 2016.

[16] C. Argyrides, D. Pradhan, T. Kocak. **Matrix Codes for Reliable and Cost Efficient Memory Chips**. *IEEE Transaction Very Large Scale Integration (VLSI) Systems*, vol. 19, n. 3, pp. 420-428, Mar. 2011.

[17] H. Castro, J. Silveira, et. al. **A Correction Code for Multiple Cells Upsets in Memory Devices for Space**. *IEEE NEWCAS*, pp. 1-6, 2016.

[18] L. Leem, H. Cho, J. Bau, Q. Jacobson, S. Mitra. **ERSA: Error Resilient System Architecture for Probabilistic Applications**. *Design Automation and Test in Europe (DATE)*, pp. 1560-1565, 2010.