Microelectronics Journal 100 (2020) 104799

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Optimized buffer protection for network-on-chip based on Error Correction Code

Alan Pinheiro [a,*], Daniel Tavares [a], Felipe Silva [a], Jarbas Silveira [a], César Marcon [b]

[a] LESC-DETI, Federal University of Ceará (UFC), Fortaleza, Brazil
[b] PPGCC, Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil

ARTICLE INFO

Keywords:
Networks-on-chip
Fault tolerance
Buffer protection
Error Correction Code

ABSTRACT

Newest technologies of integrated circuits manufacture require a communication architecture such as a Network-on-Chip (NoC). The NoC buffers are susceptible to Multiple Cell Upsets (MCU). Besides, as the technology scales down, the probability of MCU increases. Thus, applying an Error Correction Code (ECC) in NoC buffers may come as a solution for reliability issues, although increasing the design costs and requiring a buffer with higher storage capacity. This work evaluates two models of data arrangement for NoC buffers protected by three types of ECCs, preserving the protection of the storage information, and reducing the area usage and power dissipation compared to other solutions. We evaluated the performance of fault-tolerant NoC buffer schemes by applying the models on three types of ECC and measuring the buffer area, power overhead and error coverage. The experimental results show that the use of the Optimized model keeps the reliability for MCU while reducing area consumption and power dissipation in approximately 25% and 30%, respectively.

1. Introduction

The increasing development of VLSI technologies allowed integrating multiple circuits into a single System-on-Chip (SoC), providing efficient solutions for countless applications in several areas, such as entertainment, telecommunication and consumer electronics [1]. These systems require high scalability, massive parallel communication and, sometimes, strict timing constraints. Network-on-Chip (NoC) is an efficient packet-oriented communication structure [2], presenting high flexibility, scalability, and parallelism [3], fulfilling the communication requirements of such applications. The NoC architecture encompasses routers, links and Network Interfaces (NIs) to connect Processing Elements (PEs) such as processors and memories. Routers, containing control logic, crossbars and buffers, dispatch the packets sent by PEs through the NoC links. Depending on the NoC topology, global links interconnect the routers, and local links connect each PE to the NI and the NI to a router [4].

Multiprocessor SoC (MPSoC) is a multiprocessing system widely applied in a broad range of parallel applications, such as network security and data transmission. Additionally, NoC-based MPSoCs bring a significant enhancement in the area usage and power dissipation, along with higher communication speed [5].

The scale-down of recent manufacturing technologies increases the MPSoC susceptibility to a Multiple Cell Upsets (MCU) incidence, causing transient faults. MCU is induced by many sources, such as radiation, coming from cosmic emissions and electromagnetic interference [6]. The most common solution to mitigate data errors of critical applications is the use of an Error Correction Code (ECC) [7]. Buffers are critical components of an NoC, responsible for temporarily store the data traffic. A strike of a charged particle may induce information errors on these buffers, compromising the data transmission or even causing a communication crash (e.g., altering the header content of a packet, producing a deadlock situation). Cho. Leem and Mitra [8] emphasize the importance of using fault-tolerant circuits in critical parts, primarily those structures dealing with crucial system information, for providing some reliability degree to maintain the application working correctly in hostile environments.

Radetzki et al. [9] present an overview of the fault tolerance methods applied to NoCs, covering performance and cost tradeoffs of implementing ECCs in input buffers. They describe that NoCs can be affected by permanent or transient errors occurring in different network levels, such as packet routing and the flow control of each router. To detect these types of errors, the designers usually adopt in-operational detection techniques and built-in self-test circuits for transient and permanent

* Corresponding author.
E-mail address: cadore@lesc.ufc.br (A. Pinheiro).

https://doi.org/10.1016/j.mejo.2020.104799
Received 16 September 2019; Received in revised form 28 February 2020; Accepted 17 April 2020
Available online 22 April 2020

errors, respectively. Achballah, Othman and Saoud [10] investigate the problems and challenges of the emerging NoC designs. Wang et al. [11] analyze the best way to apply ECC techniques to improve the NoC reliability. They studied six protection approaches to achieve a protection strategy with the best tradeoff among delivery rate, latency, area overhead, and energy consumption.

Majumber, Suri and Shekhar [12] propose a hybrid buffer for cells of Static Random-Access Memory (SRAM) and Spin Transfer Torque-Magnetic RAM (STT-MRAM) with Single-Error Correct-Double Error Detection (SEC-DED) mechanism against transient faults. The proposal considers the buffer depth to minimize the performance impact since 4-bit cells of Single Event Effect-MRAM (SEE-MRAM) can replace an SRAM bit cell. The routers retransmit flits based on the ECC for flits received with uncorrected errors. The authors conclude that the strategy used for limited flit relay resolves the stochastic switching effect on STT-MRAM devices without affecting the NoC flow and latency.

Silva et al. [13] analyze the impact of applying some types of ECCs for protecting NoC buffers against transient faults. The implementations of both encoder and decoder of all proposed ECCs showed a low impact on the area usage and power dissipation. However, the redundancy words need to be stored in buffers, as well as transmitted inside the packet flits. They stored data and redundancy bits in a model called *extended buffer*, where for each data address, the subsequent address contains the redundancy word. This model increases the buffer width significantly; thus, rising the area susceptible to MCU.

This work extends our previous scheme for NoC buffer design [14] and analyses the impact of an optimized ECC model for implementing three types of ECCs on the router buffers: ExHamming [13], ExHamming with interleaving and FUEC-TAEC [15]. These router buffers are designed to store the packet flits that travel in the NoC, as well as the redundancy words generated by the encoders. We evaluated the proposed schemes applying a set of synthetic traffics considering (i) latency results per traffic load offered, (ii) traffic accepted per load offered, and (iii) error coverage when applying 1 to 4 error patterns. Also, we extract circuit synthesis results for area consumption and power dissipation of the NoC routers and buffers.

2. ECCs applied to the target architecture

We choose the router buffers of NoC Phoenix [16] to evaluate the proposed technique implemented in three types of ECCs. Phoenix has regular mesh topology, wormhole switching and five dual-channel ports per router. Each router implements the XY routing algorithm, on-off flow control and Round-Robin arbitration. Fig. 1 presents the router structure of NoC Phoenix, which uses circular FIFO (First-In-First-Out) buffers for each one of the five input ports: North, East, West, South, and Local.

The input and output of all buffers of NoC Phoenix contain encoders and decoders, respectively, that implement ExHamming, ExHamming with interleaving and FUEC-TAEC, with ECC optimization techniques. All mechanisms for link protection and monitoring are omitted since this work focuses only on buffer protection.

2.1. ExHamming code

Hamming is a well-known ECC developed by Richard Hamming [17] that can *correct a single error* or *detect double errors*. Hamming is one of the first ECCs applied to critical applications that require high data reliability. Equation (1) shows that Hamming encodes m data bits in n bits requiring k redundancy bits.

$$m = n - k \tag{1}$$

Additionally, Hamming defines that for each integer $n \geq 2$ there is a code size illustrated by Equation (2), which can be rewritten according to Equation (3).

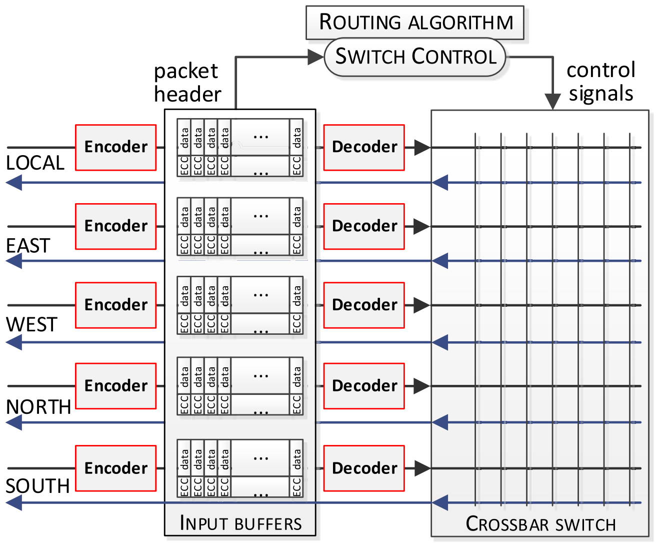


Fig. 1. Router structure of NoC Phoenix, highlighting the encoders, decoders and input buffers [14].

$$n = 2^k - 1 \qquad \forall n \geq 2 \tag{2}$$

$$k = \log_2(n+1) \qquad \forall n \geq 2 \tag{3}$$

Equation (3) shows that Hamming is a low-cost ECC, where the redundancy bits grow proportionally to $\log_2(n)$. ExHamming extends the original Hamming code by adding a parity bit to reach the SEC-DED capacity while maintaining the same low-cost characteristic.

2.2. ExHamming code with interleaving

The reasoning for using the interleaving method is to enforce that double errors or even more complex errors occur in different words. The appliance of interleaving architectures, combined with ECCs, provides a significant increase in reliability for data information [18].

Fig. 2 presents the interleaving pattern applied in this work in three steps. The interleaving process allows correcting double or more errors for ECCs with low correction capability, like the Hamming code. Step 1 splits a 16-bit of data (D) into two 8-bit of data composing the words A and B. Step 2 encodes each word separately according to the ExHamming code, producing the codewords A′ and B'. Note that ExHamming codifies 8 data bits with more 5 redundancy bits, whereas 16 data bits require only more 1 bit of redundancy, demonstrating the increase in redundancy when applying ExHamming with interleaving. Finally, Step 3 interleaves the codewords A′ and B′ producing AB'. The use of ExHamming with interleaving increases the correction and detection capability of the standard ExHamming, allowing reducing the occurrence of adjacent error patterns.

2.3. FUEC-TAEC code

Gracia-Morán et al. [15] proposed the Triple Adjacent Error Correction with Flexible Unequal Error Control Methodology (FUEC-TAEC) as a new ECC. FUEC-TAEC is a linear code that detects and corrects data errors applying a parity-check matrix H [19], which is computed using the Boolean Satisfiability problem through a recursive backtracking algorithm. FUEC-TAEC has, as the main advantage, the high correction capability against adjacent errors and low redundancy usage.

The authors explain in Refs. [15] that after defining the values of n and k (the same parameters of Equation (1)), the first step is the selection of error patterns to be corrected and detected. The next step is to find the matrix H that satisfies Equations (4) and (5), where e is the error vector, E_+ is the set of vectors to be corrected, and E_Δ is the set of errors that can

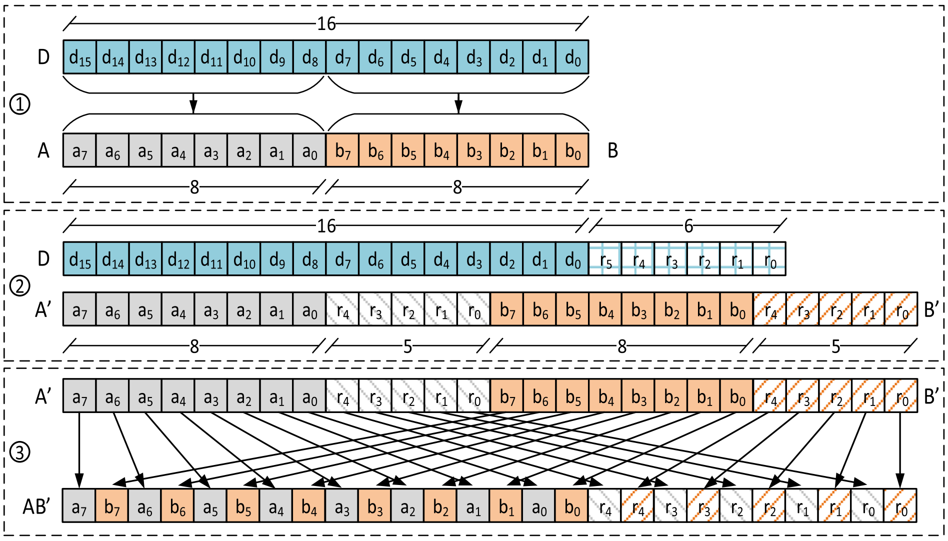


Fig. 2. Example of applying the ExHamming with interleaving code in a 16 data bit.

be detected. It means that each correctable error must generate a different syndrome, and each detectable syndrome must generate a new syndrome, being different to all syndromes generated by correctable errors.

$$H \times e_i^T \neq H \times e_j^T \qquad \forall e_i, e_j \in E_+ \big| e_i \neq e_j \tag{4}$$

$$\mathrm{H} \times \mathrm{e}_\mathrm{i}^\mathrm{T} \neq H \times e_i^T \qquad \forall e_i \in E_\Delta, e_j \in E_+ \tag{5}$$

2.4. Summary of the ECCs

Table 1 summarizes the characteristics of the ECCs presented in this section, considering 16 data bits; this table shows that each ECC has a specific size for the redundancy words. Additionally, Table 1 shows the capacity of detecting and correcting errors of each ECC.

3. Extended buffer scheme

We implemented ExHamming, ExHamming with interleaving, and FUEC-TAEC codes using (i) *Basic*, (ii) *Extended*, and (iii) *Optimized* buffer schemes. The *Basic* buffer scheme is a straightforward buffer implementation; the *Extended* buffer approach was proposed by Silva et al. [13], whereas the *Optimized* buffer scheme was proposed in Ref. [14]. Our previous work [14] only applied the *Optimized* scheme for ExHamming code; here, we extend this model assessment by adding two different ECCs: Interleaved ExHamming and FUEC-TAEC codes.

This section describes the *Extended* buffer scheme, while the next section explains the *Optimized* one.

Let *Basic* buffer be a buffer implemented without redundancy; then, the structure of the *Extended* buffer requires expanding the width of the *Basic* buffer according to the redundancy word generated by a given ECC. Fig. 3 shows that this approach encodes each data-flit on every pushing

Table 1
ECC structures for 16 data bits, together with their error correction and detection coverages.

ECC	Width (bits)			Coverage	
	Redundancy	Data	Total	Correction	Detection
ExHamming	6	16	22	1	2
ExHamming + Interleaving	10	16	26	2	4
FUEC-TAEC	8	16	24	3	4

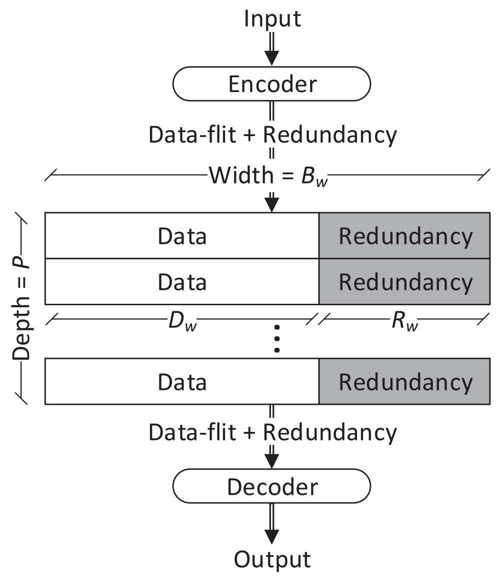


Fig. 3. Data flow and structure of the *Extended* buffer scheme.

procedure before writing into the buffer. The redundancy word is written side-by-side with its related data-flit in the same buffer address. The pulling procedure decodes the data after reading the buffer address; next, this procedure transmits the data to the output link. Each access is performed in one clock cycle.

Equation (6) defines the buffer width (B_w) of the *Extended* scheme. Where D_w is the data width and R_w is the width of the redundancy word, which changes according to the applied ECC.

$$B_w = D_w + R_w \tag{6}$$

Let P be the buffer depth; then, Equation (7) computes the total of bits in the buffer (T_b).

$$T_b = P \times B_w \tag{7}$$

4. *Optimized* buffer scheme

This work defines *real depth* and *effective depth* concepts to explain the *Optimized* buffer approach. The first concept describes the total number

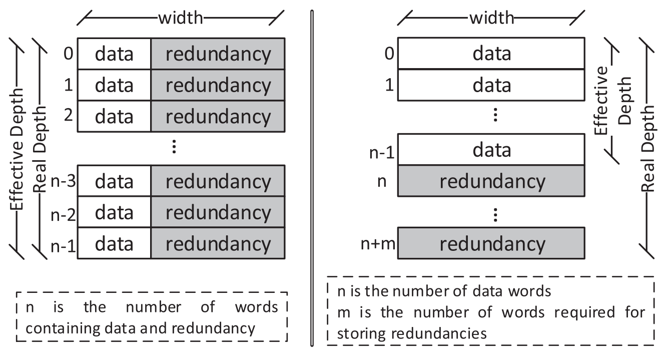


Fig. 4. Concepts of *real depth* and *effective depth* applied to the two schemes of buffers.

of addresses that a buffer has, i.e., it is the number of buffer addresses in the physical implementation. The second concept is the number of addresses available for data-flit storing, i.e., without regarding redundancy addresses. Fig. 4 shows these concepts. Note that the *effective depth* can be equal or less than *real depth*, but never higher.

The *Extended* buffer scheme uses *effective depth* equals to the *real depth*; thus, the buffer width must be larger than the *Basic* buffer. Differently, the *Optimized* buffer approach keeps the buffer width identical as the *Basic* buffer scheme and sets the *effective depth* less than the *real depth*. The remaining addresses are reserved for the redundancy words; consequently, increasing the data and redundancy regions.

The *Optimized* buffer scheme requires specific circuits for each ECC to avoid bit fragmentation during the data positioning into the buffer. We defined Equations (8)–(11) based on the Least Common Multiple (LCM) between D_w and R_w to find the Minimum storing Relation of Data and Redundancy (MRDR). Equation (8) computes the number of addresses for redundancy words (A_r), and Equation (9) calculates the number of addresses for data words (A_d). Note that $A_d = n$, whereas $A_r = m$. Equation (10) computes the real depth (P) by $A_r + A_d$. Finally, Equation (11) calculates the total of bits (T_b).

$$A_r = \frac{LCM(D_w, R_w)}{D_w} \tag{8}$$

$$A_d = \frac{LCM(D_w, R_w)}{R_w} \tag{9}$$

$$P = LCM(D_w, R_W) \times \left(\frac{D_w + R_w}{R_w \times D_w}\right) \tag{10}$$

$$T_b = LCM(D_w, R_W) \times \left(\frac{D_w + R_w}{R_w}\right) \tag{11}$$

4.1. ExHamming buffer architecture

Applying Equations (8)–(10) with 16 data bits, ExHamming has MRDR defined by $A_r = 3$, $A_d = 8$, and $P = 11$; i.e., the data region has 8 addresses (*effective depth*), the redundancy region has 3 addresses, and thus, the entire buffer has 11 addresses (*real depth*). Fig. 5 shows that this technique segments some redundancy words into two buffer addresses to take advantage of buffer space.

R_2 is an example of a redundancy segmentation; the addresses 8 and 9 store the first (4 bits) and second (2 bits) parts of R_2, respectively. This scheme implies that the data-redundancy logical pair uses 2 to 3 addresses depending on the segmentation of the redundancy word. Additionally, the logical pair data-redundancy is correlated by indexes; e.g., when a data is stored into the address 0 (data region), the associated redundancy is R_0 (redundancy region).

Fig. 6 illustrates the data flow of the *Optimized* buffer from encoding until decoding. In a single clock cycle, the encoder codifies every incoming data into a data-redundancy logical pair and writes the data and redundancy into the next available data region and redundancy region addresses, respectively. These addresses are pointed by the *Head* and *Tail* registers that perform the write and read FIFO positions, respectively.

While the data storage requires a simple binary decoder to select the buffer positioning, the redundancy storage requires two circuits. The *Address Generator* circuit selects the buffer address based on the *Head* pointer, and the *Bit Mask* circuit places the redundancy word in the specific bit position of the buffer (which can be segmented or not).

The reading process is similar to the writing process. In a single clock cycle, the decoding circuit reads the next available data-redundancy pair addressed by the *Tail* pointer. Then, the decoder evaluates and eventually corrects the data according to the decoding pattern.

4.2. Interleaved ExHamming application

Applying Equations (8)–(10) with 16-bit data, the MRDR for Interleaved ExHamming is $A_r = 5$, $A_d = 8$, and $P = 13$. Fig. 7 shows that these parameters require to segment 4 redundancy words to fit into this configuration, resulting in four fragmentation sizes. In the Interleaved ExHamming scheme, there are more segmented words than the previous one; therefore, the data access requires a more complex circuit.

4.3. FUEC-TAEC application

Fig. 8 illustrates that FUEC-TAEC employs a regular distribution of

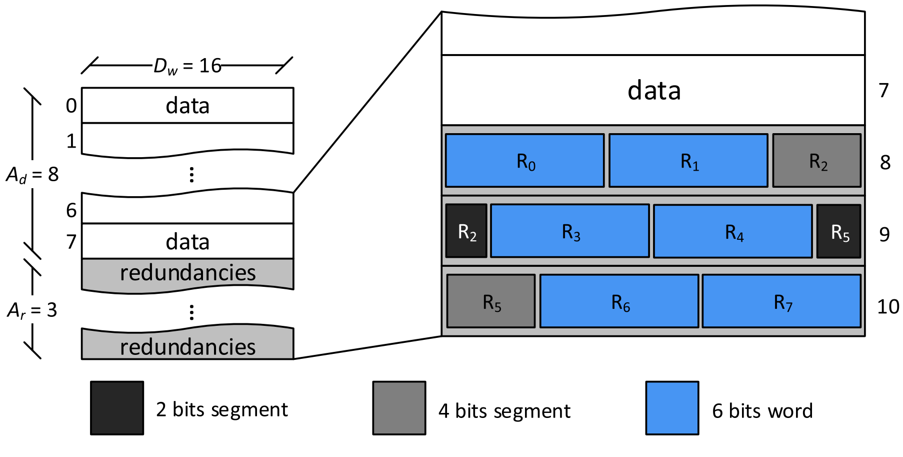


Fig. 5. Data and redundancy positioning for the buffer architecture implementing ExHamming.

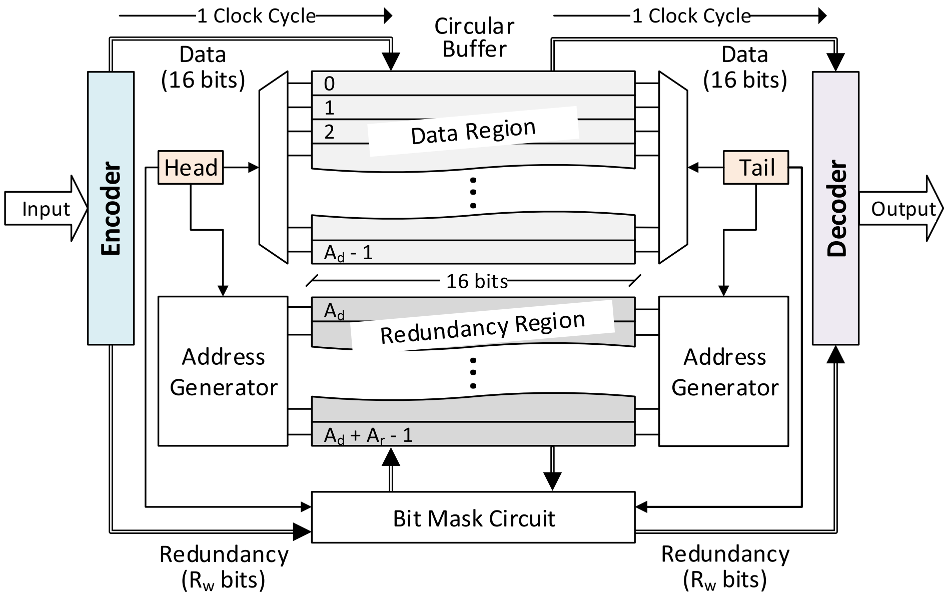


Fig. 6. Architectural model of the *Optimized* buffer scheme of a router input buffer, and the data flow, which encompasses encoding, storage, decoding and address generation.

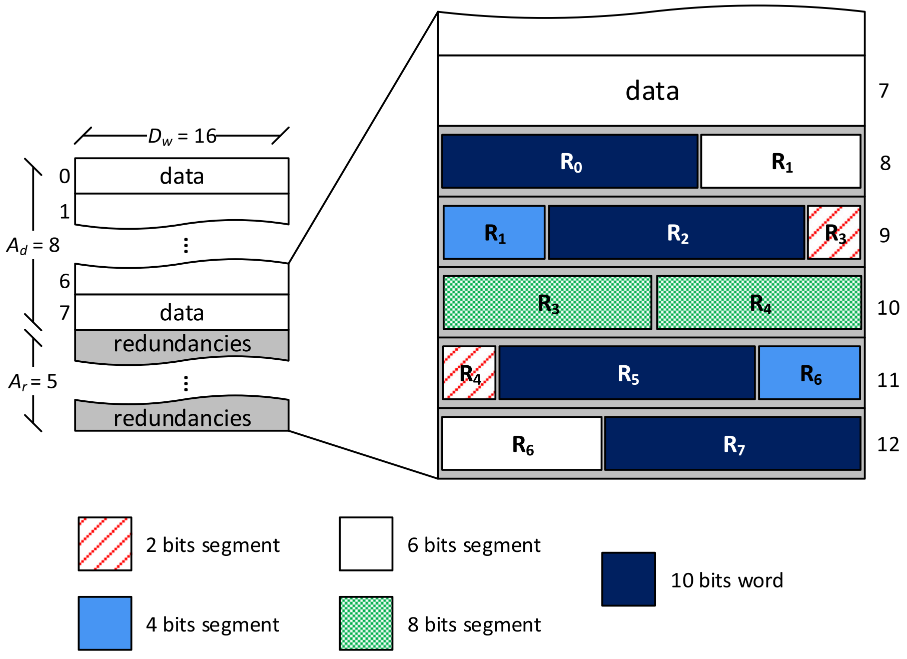


Fig. 7. Data and redundancy positioning for the buffer architecture with Interleaved ExHamming code.

the redundancy words; as D_w is the double of R_w, it is not necessary to segment any redundancy word. Applying Equations (8)–(10) with 16 data bits, FUEC-TAEC has MRDR defined by $A_r = 1, A_d = 2$, and $P = 3$. However, for a closer dimension to the other schemes, we defined A_d with the same value of the previous architectures ($A_d = 8$). Consequently, the other variables were defined as $A_r = 4$ and $P = 12$. These variables configure the dataflow model demonstrated in Fig. 6.

4.4. Theoretical comparison

Equations (7) and (11) provide an analytical formulation to compare and estimate which model has the highest area consumption. To do this, we analyzed the number of bits each model required. The larger the R_w, the larger the buffer of the *Extended* scheme. Also, R_w specifies the configuration of A_d and A_r for the *Optimized* buffer approach. Let P' and P'' be the depths of *Optimized* buffer and *Extended* buffer approaches, respectively; let $T_b^{'}$ and T_b'' be the total of bits of *Optimized* buffer and *Extended* buffer, respectively; then, setting $P' = P''$ makes $T_b^{'} < T_b^{''}$ for any ECC applied.

5. Synthesis results

This section presents and discusses the results of the *Basic*, *Extended* and *Optimized* schemes synthesized with Cadence's *Genus* for 65 nm CMOS technology. Fig. 9 to Fig. 11 show the impacts of *Optimized* and *Extended* schemes on ExHamming, ExHamming with interleaving and FUEC-TAEC ECC architectures. For comparison purposes, we use the *Basic* buffer scheme as a reference, which has the same *real depth* as the

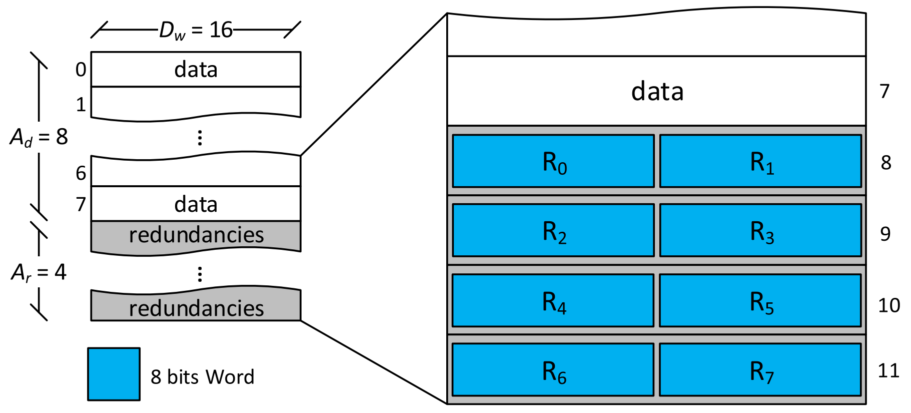


Fig. 8. Data and redundancy positioning for the buffer architecture of the FUEC-TAEC code.

other schemes.

For each ECC application, the encoders and decoders of the *Extended* and *Optimized* schemes are the same. However, the *Optimized* scheme has a more specialized access circuit for the data buffer. Other differences are the total of bits that depends on the coding architecture (defined in Equation (11)) and the redundancy alignment.

Fig. 9 shows that ExHamming consumes less area than Interleaved ExHamming because the interleaving process requires two 8-bit data encoders-decoders (refer to Fig. 2). Although the *Optimized* scheme saves area, the interleaving approach implies more power dissipation for both *Optimized* and *Extended* schemes. The *Extended* and *Basic* buffer schemes have the same generic data access based on the FIFO mechanism, whose implementation complexity is higher than a specialized circuit of the *Optimized* scheme. The *Extended* scheme has more area than the *Optimized* scheme because of the ECC implementation and the number of storage bits (Equation (7)). FUEC-TAEC has the lowest cost in the *Optimized* scheme because it has a more straightforward logic of data access; besides, the *Optimized* and *Basic* schemes have the same number of storage bits (Equation (11)).

The *Optimized* buffer scheme simplifies the data access, having specific implementations for each ECC (Equations (8)–(11)). Fig. 9(b) shows that the values for the *Optimized* scheme of FUEC-TAEC are smaller due to the simplicity of the data layout, which lets the data access circuit even simpler.

Fig. 10 shows that the larger area of the Interleaved ExHamming circuit implies an increase in the static energy dissipation when compared to the ExHamming architecture. Additionally, Fig. 10 illustrates the leakage power of the *Optimized* buffer is lower than the *Extended* buffer for all architectures because the *Optimized* scheme is more cost-effective than the *Extended* scheme, mainly when implemented in the FUEC-TAEC code.

Fig. 11 shows the dynamic power of all architecture, revealing the activity of each circuit. The *Extended* and *Optimized* schemes dissipate more dynamic power than the *Basic* buffer, due to the coding and decoding activity.

The *Optimized* buffer scheme is more efficient than the *Extended* buffer scheme in the *ExHamming* architecture, although both solutions have the same encoder-decoder circuits. The *Interleaved ExHamming* architecture reveals that both solutions have nearly the same dissipation because the amount of dynamic power produced by the ECC is equally high and prevalent. Finally, FUEC-TAEC coding/decoding requires a simpler data access circuit; thus, minimizing the dynamic power dissipation.

6. Error coverage analysis

Fig. 12 illustrates the experimental setup employed to analyze the

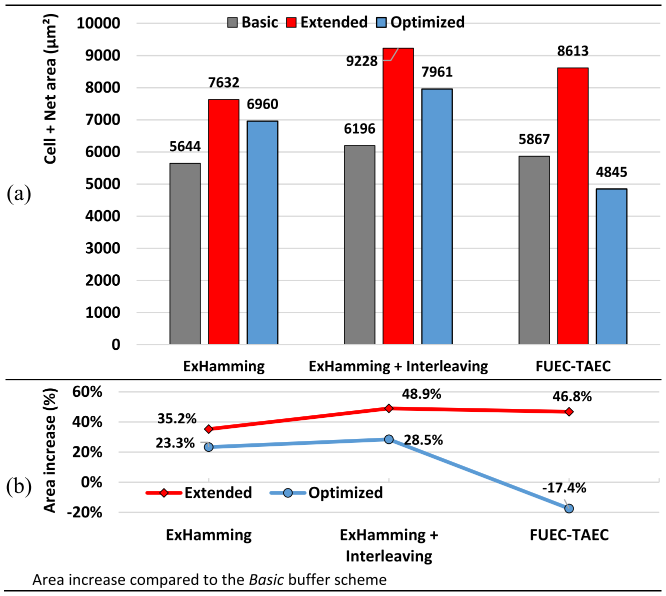


Fig. 9. (a) Absolute and (b) relative area consumption of the three ECCs implemented in *Extended* and *Optimized* approaches.

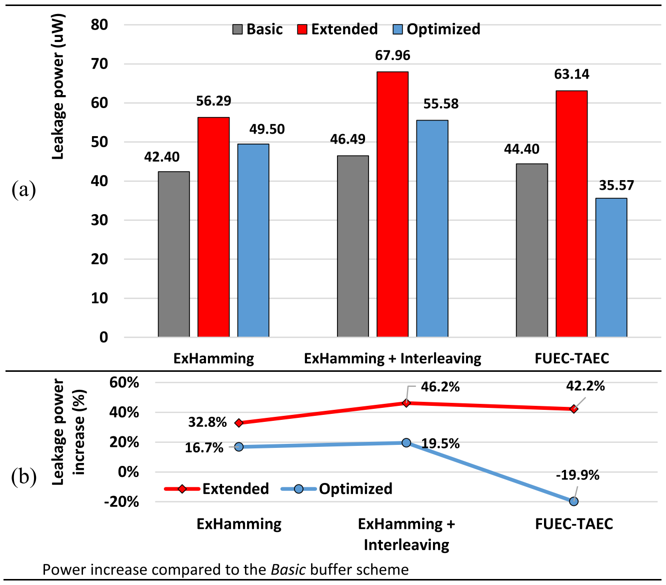


Fig. 10. (a) Absolute and (b) relative leakage power dissipation of the ECCs implemented in *Extended* and *Optimized* approaches.

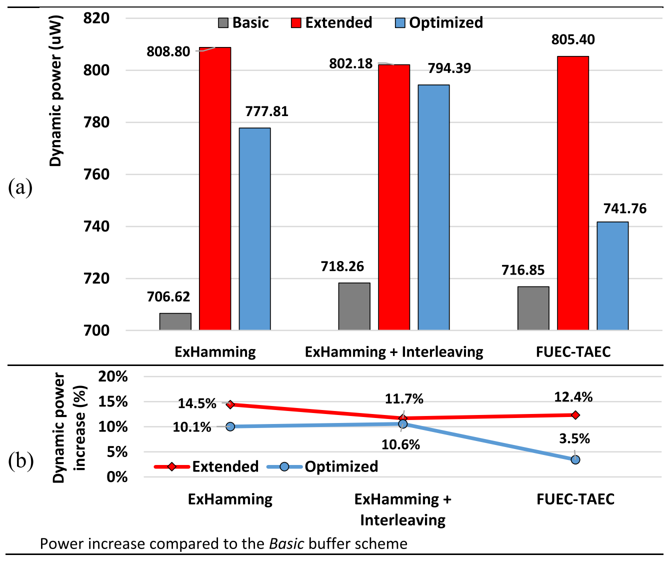


Fig. 11. (a) Absolute and (b) relative dynamic power dissipation of the ECCs implemented in *Extended* and *Optimized* approaches.

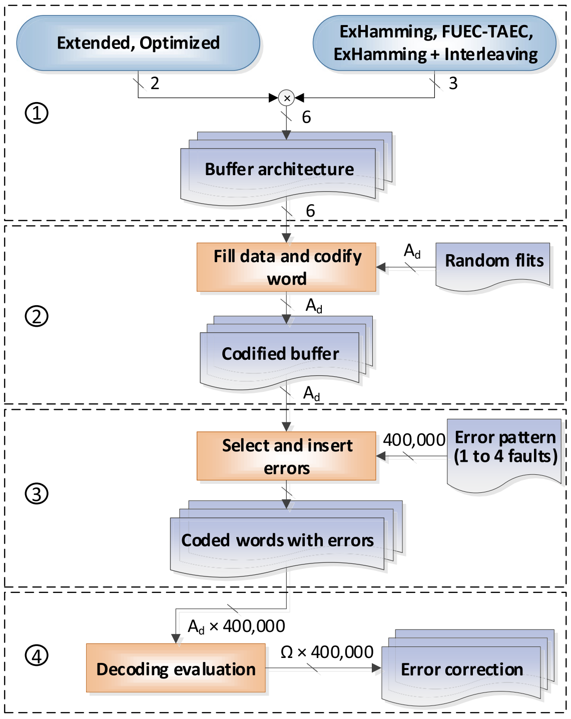


Fig. 12. Experimental setup for evaluating the techniques proposed in this work.

error coverage of all implementations proposed here. We guide this analysis considering the correction rate as the robustness metric.

The factors of study are the protection level that changes between each ECC, and the error patterns injected into the buffer, which varies the error severity degree. The *Basic* buffer scheme is not included here because it does not implement ECC circuits.

Step 1 combines the two schemes of buffer protection with the three implemented ECCs, resulting in 6 solutions. Next, Step 2 defines the writing data procedure; the current ECC circuit encodes each input flit and stores with its redundancy word. This analysis is performed in a single buffer using flits with random data.

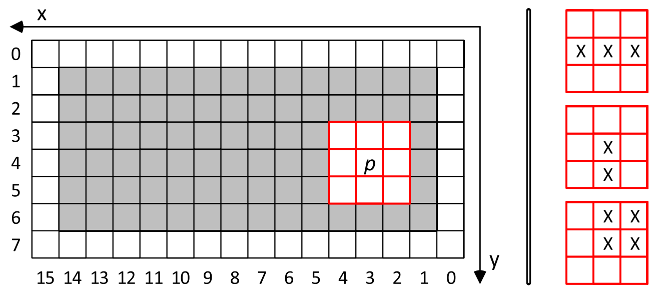


Fig. 13. Proceeding of error injection in a synthetic 16 × 8 buffer.

Step 3 injects into the buffer one error pattern per analysis. For all analyses, we injected from 1 to 4 adjacent errors in random positions, based on the experiments described in Ref. [20]. First, we inject 100,000 errors of 1-bit flip and proceed to Step 4. The storage of bits is spatially positioned as a memory in the injection proceeding. All error patterns are distributed in a 3 × 3 matrix; the right side of Fig. 13 illustrates samples of error patterns. Fig. 13 shows an error pattern being injected in a random position *p(x,y)*, which can be any coordinate of the gray region.

Each decoding operation is monitored during the reading procedure in Step 4, and all correction/detection results are recorded. Ω represents the unknown set of counted corrections per error injection. Steps 2, 3 and 4 are executed 100,000 per error severity (1, 2, 3 and 4 errors per pattern) for each architecture. Therefore, the total amount of error patterns injected is 400,000.

Table 2 shows the error coverage results for all buffer architectures in the *Extended* and *Optimized* schemes. These results indicate that the *Optimized* scheme is equally efficient for error correction as the *Extended* scheme. The 3 and 4 errors are out of the ExHamming error correction/detection capacity because ExHamming is an SEC-DED. On the other hand, the Interleaved ExHamming code increases this correction/detection capacity since the patterns contain adjacent errors that are placed in different codewords.

Fig. 14 depicts that some patterns of adjacent errors may be in different data and redundancy cells, and/or positioned at the border that divides the data and redundancy region (striped region), allowing the ECC to correct these types of errors.

ExHamming reaches a high correction rate for two errors because the used error pattern makes most errors occurring in different words; i.e., although a single ExHamming code can fix only one error, the error pattern implies corrections with more than one cell. Fig. 15 shows one of these occurrences with a pattern consisting of two errors that affect the bit 5 of words A and B.

As errors become more aggressive (i.e., 3 or 4 errors), the occurrence of double errors in the same data word increases, reducing the correction rate. Note that the ExHamming is still able to correct 3-error and some 4-error patterns. Therefore, a 4-error pattern can be distributed in 4 different words.

Considering that the correction capacity of Interleaved ExHamming is potentially the double of ExHamming, it is expected that the patterns with 1 and 2 errors (Table 2) always be corrected. In this evaluation, it was also verified that the *Optimized* scheme discreetly surpasses the *Extended* one for corrections performed in patterns with 3 and 4 errors.

FUEC-TAEC should correct all cases of errors inserted into the buffer. However, for 3 errors, the correction rate is not 100%. This behavior is justified by the existence of 3 error patterns that provoke two non-adjacent errors in one data, and the third one hits another data. Data B and C demonstrate this situation in Fig. 16.

The 4-error pattern experiment with FUEC-TAEC shows that its corrections stand out the Interleaved ExHamming code. This is a consequence of some 4-error patterns contain three adjacent errors in their spatial positioning, which is a correctable situation for the FUEC-TAEC code.

Table 2
Error coverage rates for all ECC architecture implementing the two fault-tolerant buffer schemes.

Number of errors	Correction rate (%)					
	ExHamming		Interleaved ExHamming		FUEC-TAEC	
	Extended	Optimized	Extended	Optimized	Extended	Optimized
1	100.00	100.00	100.00	100.00	100.00	100.00
2	75.01	76.07	100.00	100.00	100.00	100.00
3	44.37	46.48	87.50	88.63	90.91	91.92
4	1.88	3.14	51.52	52.25	60.01	59.95

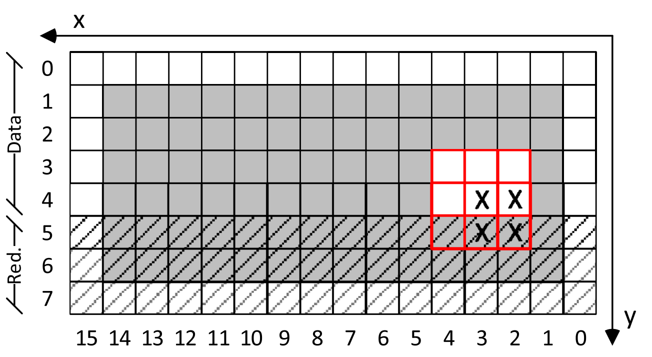
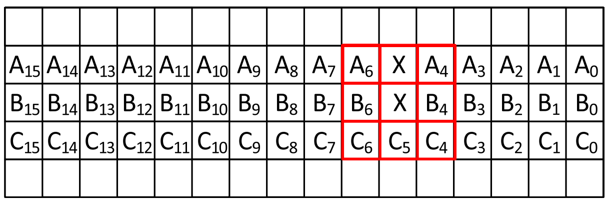
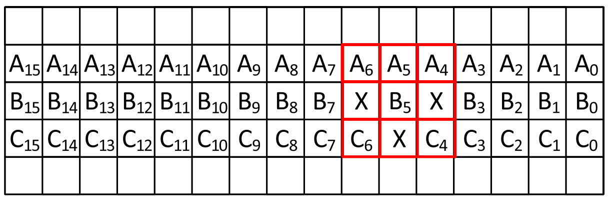


Fig. 14. Error pattern hitting data and redundancy regions of a synthetic 16 × 8 buffer.


A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 X A4 A3 A2 A1 A0
B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 X B4 B3 B2 B1 B0
C15 C14 C13 C12 C11 C10 C9 C8 C7 C6 C5 C4 C3 C2 C1 C0
Fig. 15. Pattern with two errors hitting two words.


A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
B15 B14 B13 B12 B11 B10 B9 B8 B7 X B5 X B3 B2 B1 B0
C15 C14 C13 C12 C11 C10 C9 C8 C7 C6 X C4 C3 C2 C1 C0
Fig. 16. Error pattern without three adjacent errors in the same line.

7. Performance analysis

This section describes the experiments of performance using the average latency and accepted traffic as metrics. The *Basic* buffer scheme was not included because it has the same performance as the *Extended* buffer scheme. The experiments explore (i) four synthetic traffic patterns; (ii) injection rates varying from 1% to 100%; and (iii) buffer depths that depend on the buffer scheme. The discussion is divided by traffic patterns as follows. The injection rates were displayed in distinct ranges because of the different saturation points. Additionally, each plot highlights the variation between the buffer schemes.

7.1. Uniform traffic

The *uniform* traffic gives routers the same probability of being a communication target [21]. Fig. 17 illustrates the impact of the buffer depth variation between the *Extended* and *Optimized* schemes when the uniform traffic pattern is applied. The saturation point appears between 30% and 40% in the accepted traffic plot. This behavior follows the increasing average latency.

The variation in every *Optimized* scheme reveals that the Interleaved ExHamming has the highest latency variation, but not very far from the others. This latency variation happens because this architecture has the highest buffer proportion denoted by $A_d = 8$ and $A_r = 5$.

7.2. Complement traffic

The *complement* traffic guides a source node to send messages to the target node placed in a complementary address [21]. For instance, supposing a 2D-NoC composed of 16 nodes addressed by four bits, the complement of address 1011 is 0100. Fig. 18 depicts that the appliance of ExHamming with interleaving has the most considerable gap between *Optimized* and *Extended* schemes. Fig. 18(c) illustrates that in the context of *complement* pattern, the saturation point of the *Optimized* scheme happens in 28%, which is earlier than the *Extended* scheme (34%). Considering that the *complement* traffic has some oscillations, the *Extended* and *Optimized* schemes have a low variation in each latency and accepted traffic results after the saturation point.

7.3. Perfect shuffle traffic

The *perfect shuffle* traffic makes a source node to send messages to a target node placed in the address that corresponds to a one-bit rotation of the source node (considering a binary address) [21]. This type of traffic models the communication behavior of some applications like Fast Fourier Transform and some sorting algorithms.

Fig. 19 illustrates that this traffic had the lowest impact on the *effective depth* variation. Thereby, Fig. 19(a, b and c) illustrate that each saturation point happens in 42% of the injection rate and the average latency are similar for all cases besides the oscillation.

7.4. Hotspot traffic

The *hotspot* traffic fixes only one target for all sources [21]. The hotspot is the most severe traffic because of only one target is defined for all source nodes, except the target node itself. Therefore, the saturation points occur in the very first injection rates (before 6%), as Fig. 20 shows in the accepted traffic plots. Consequently, the average latency increases very quickly compared to the other traffics. Also, the increasing curve is very similar between the buffer schemes, aside from the oscillations.

Given these results, we can confirm that when the buffer increases in effective depth, the saturation point must increase. However, the application has also an influence on the packet latency and network throughput.

Table 3 resumes the variation average between *Extended* and *Optimized* schemes for all ECCs when using the four traffic patterns explored here.

The *perfect shuffle* and *hotspot* patterns did not have considerable influence on the variation of the *effective depth*; none of them reaches 1% of the variation in latency nor accepted traffic. The *uniform* and *complement* traffics have more communication pairs, so there is further resource concurrence, and congestions increase latency.

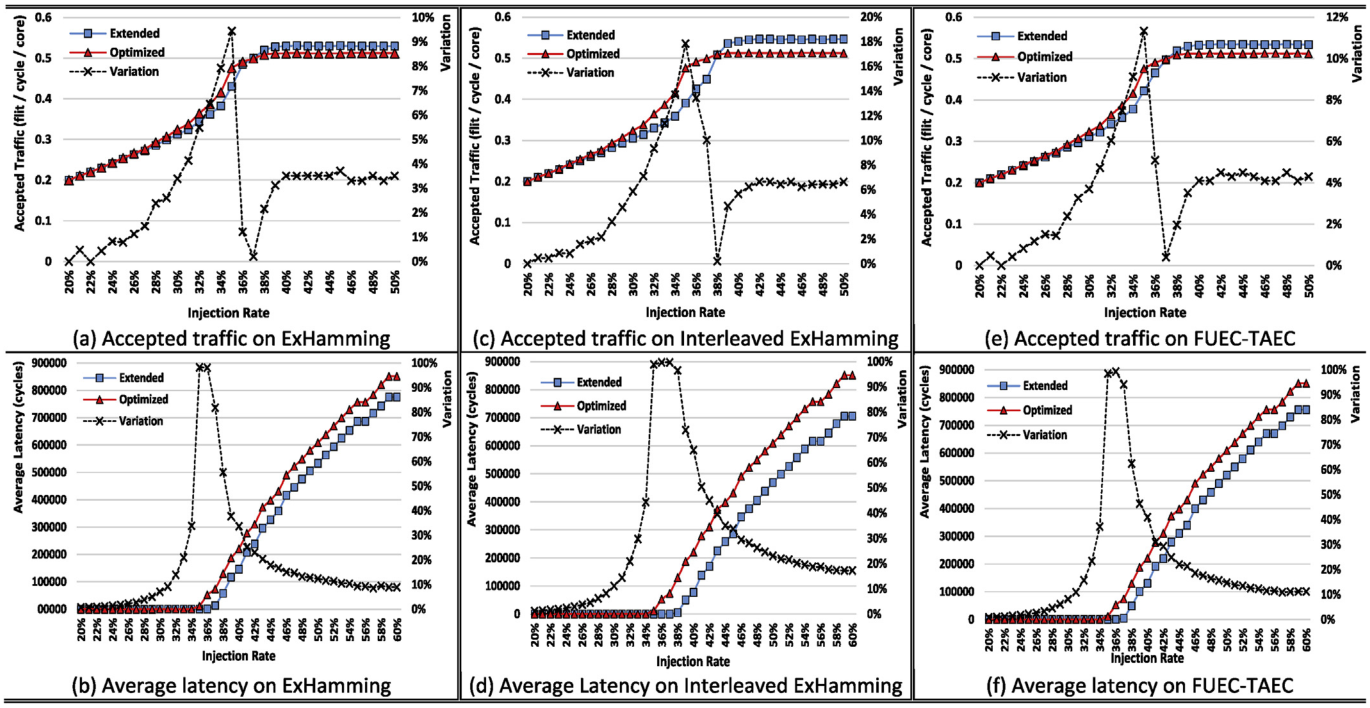


Fig. 17. UNIFORM distribution of the accepted traffic and average latency for *Extended* and *Optimized* buffer schemes of the three evaluated architectures: (a, b) ExHamming, (c, d) Interleaved ExHamming and (e, f) FUEC-TAEC.

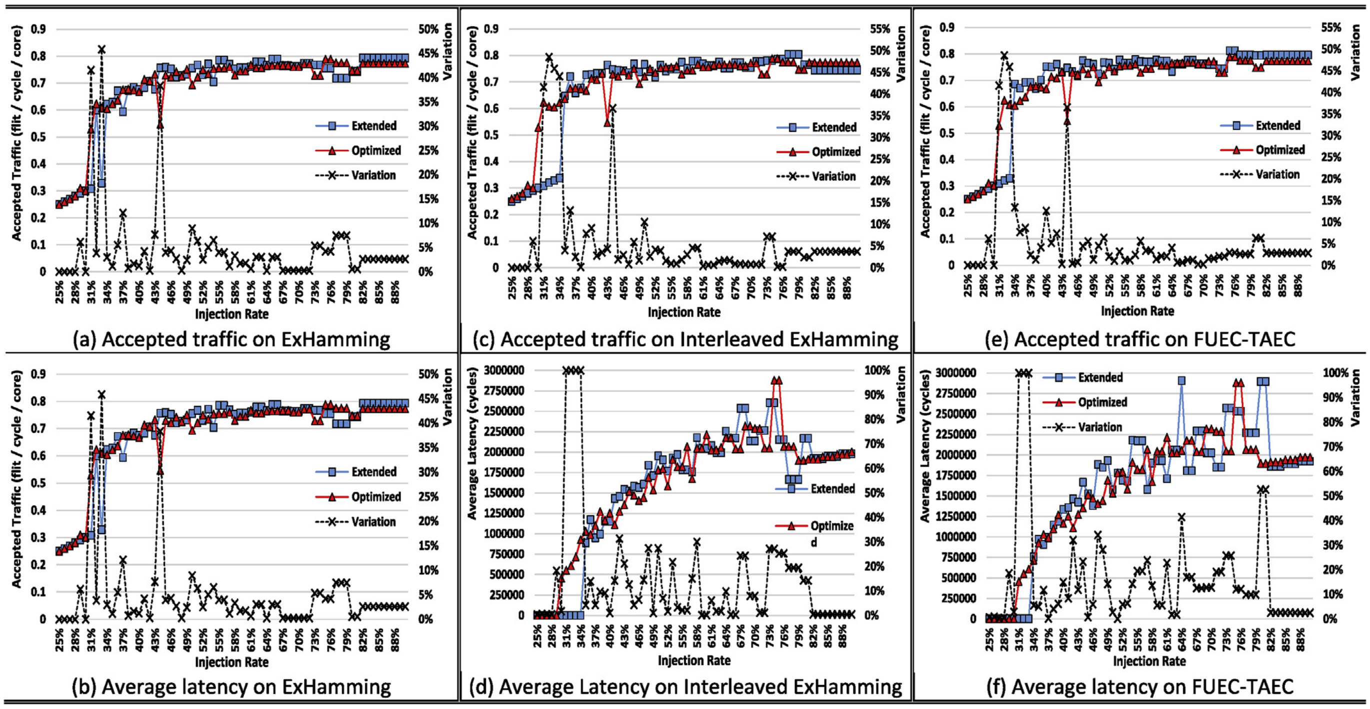


Fig. 18. COMPLEMENT distribution of the accepted traffic and average latency for *Extended* and *Optimized* buffer schemes of the three evaluated architectures: (a, b) ExHamming, (c, d) Interleaved ExHamming and (e, f) FUEC-TAEC.

8. Conclusion

This work evaluates the *Optimized* [14] and *Extended* [13] buffer schemes of fault-tolerant NoC buffer architectures for applying three ECCs of different correction levels. The *Optimized* buffer scheme applies protection to all stored data. The cost analysis in the synthesis showed that the *Optimized* buffer scheme presents better results than the *Extended* one in terms of area usage and power dissipation.

Error coverage experiments revealed that the *Optimized* scheme overcomes the *Extended* buffer scheme in almost all error patterns. The internal redundancy words positioning in the buffer raises the possibility of an MCU strike more than one data word; consequently, each bit flip tends to be corrected by different decoders.

The *Optimized* buffer scheme reduces the area consumption and

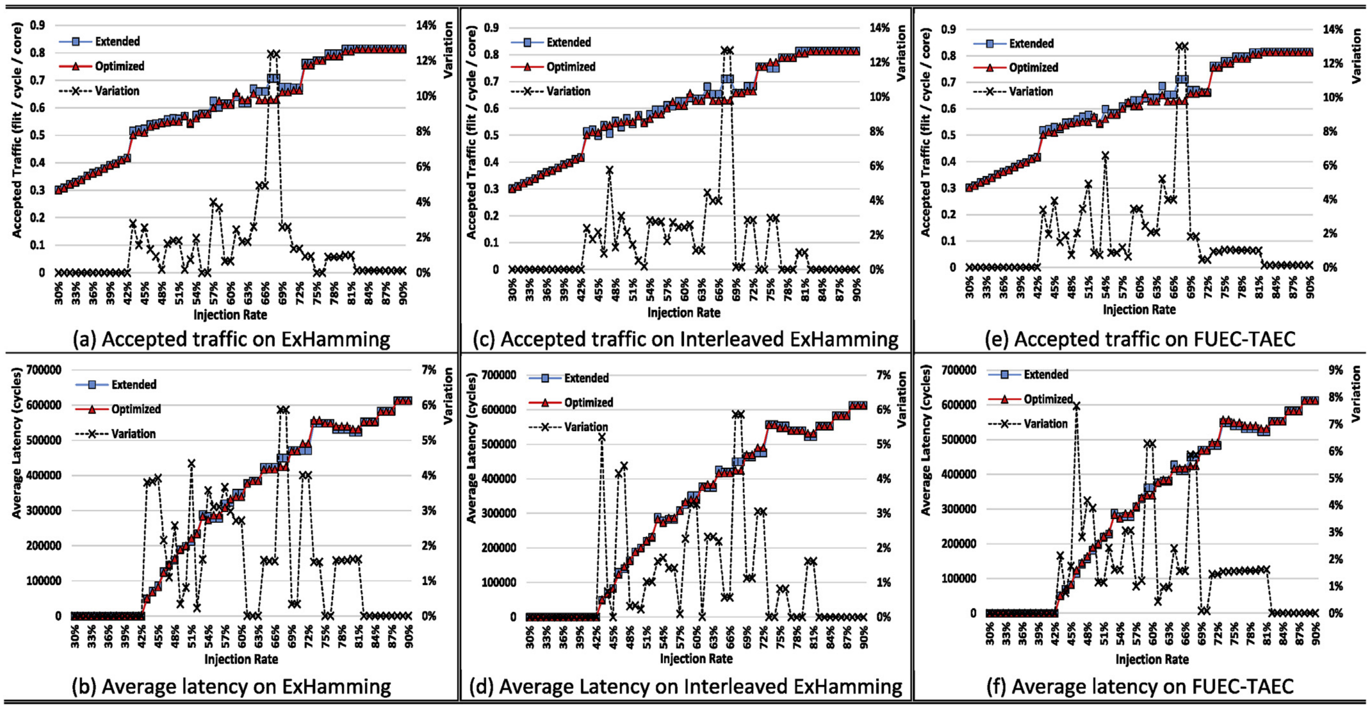


Fig. 19. PERFECT SHUFFLE distribution of the accepted traffic and average latency for *Extended* and *Optimized* buffer schemes of the three evaluated architectures: (a, b) ExHamming, (c, d) Interleaved ExHamming and (e, f) FUEC-TAEC.

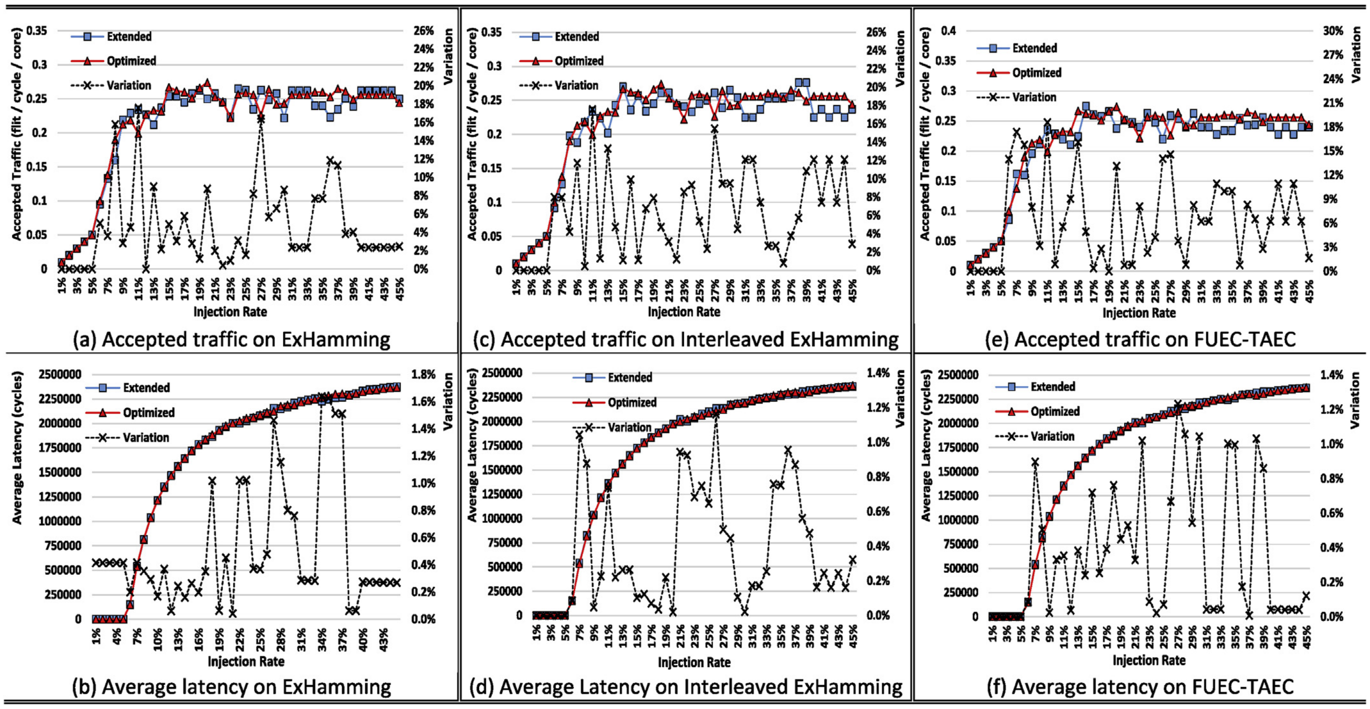


Fig. 20. HOTSPOT distribution of the accepted traffic and average latency for *Extended* and *Optimized* buffer schemes of the three evaluated architectures: (a, b) ExHamming, (c, d) Interleaved ExHamming and (e, f) FUEC-TAEC.

power dissipation compared to the *Basic* buffer scheme applying the FUEC-TAEC and provides similar reliability when compared to the *Extended* scheme. Additionally, the performance analysis showed that there is a low impact between *Extended* and *Optimized* buffer schemes, generating a small variation in saturation points in each synthetic traffic applied. Therefore, the *Optimized* buffer scheme becomes a more exciting solution for implementing ECC buffers than the *Basic* and *Extended* buffer schemes.

Finally, it is essential to note that the size of the redundancy word generated by an ECC has a direct impact on network latency. This impact depends on the *Optimized* scheme configuration (i.e., effective depth). The size of this word also directly interferes in the complexity of data access, even if the area and power costs are acceptable. For instance, the FUEC-TAEC results show that the Optimized scheme performs better

Table 3
Summary of variation between *Extended* and *Optimized* schemes for each ECC and each traffic pattern.

Mean of variation between *Extended* and *Optimized* buffer schemes (%)								
ECC	Uniform		Complement		Perfect shuffle		Hotspot	
	Lat	AT	Lat	AT	Lat	AT	Lat	AT
ExHamming	10.51	2.72	9.19	3.38	0.83	0.86	0.37	3.41
Interleaved ExHamming	17.36	5.10	10.15	4.36	0.65	0.96	0.32	5.69
FUEC-TAEC	12.20	3.27	11.27	3.89	0.89	1.04	0.21	5.02
Average	13.36	3.70	10.21	3.88	0.79	0.95	0.30	4.71

Lat – Average Latency.
AT – Accepted Traffic.

when the ECC redundancy word has a half-size of a flit. This conclusion can guide the developer to select a more efficient ECC to reach the best cost-effective in future works.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Alan Pinheiro: Methodology, Formal analysis, Writing - original draft. **Daniel Tavares:** Methodology, Formal analysis, Writing - original draft. **Felipe Silva:** Methodology, Formal analysis, Writing - original draft. **Jarbas Silveira:** Methodology, Formal analysis, Writing - original draft. **César Marcon:** Methodology, Formal analysis, Writing - original draft.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] L. Benini, G. De Micheli, Networks on chips: a new SoC paradigm, Computer 35 (1) (Jan. 2002) 70–78.
[2] S. Jiang, Y. Liu, J. Luo, H. Cheng, G. Luo, Study of fault-tolerant routing algorithm of NoC based on 2D-mesh topology, in: IEEE International Conference on Applied Superconductivity and Electromagnetic Devices (ASEMD), 2013, pp. 189–193.
[3] A. Salaheldin, K. Abdallah, N. Gamal, H. Mostafa, Review of NoC-based FPGAs architectures, in: IEEE International Conference on Energy Aware Computing Systems & Applications (ICEAC), 2015, pp. 1–4.
[4] S. Khan, S. Anjum, U. Gulzari, F. Torres, Comparative analysis of network-on-chip simulation tools, IET Comput. Digital Tech. 12 (1) (Sep. 2017) 30–38.
[5] W. Wolf, A. Jerraya, G. Martin, Multiprocessor system-on-chip (MPSoC) technology, IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. (TCAD) 27 (10) (Oct. 2008) 1701–1713.
[6] P. Ferreyra, C. Marques, R. Ferreyra, J. Gaspar, Failure map functions and accelerated mean time to failure tests: new approaches for improving the reliability estimation in systems exposed to single event upsets, IEEE Trans. Nucl. Sci. 52 (1) (Feb. 2005) 494–500.
[7] S. Lin, D. Costello, Error Control Coding, second ed., Pearson-Prentice Hall, 2004, p. 1272.
[8] H. Cho, L. Leem, S. Mitra, ERSA: Error resilient system Architecture for probabilistic applications, IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. (TCAD) 3 (4) (Apr. 2012) 546–558.
[9] M. Radetzki, C. Feng, X. Zhao, A. Jantsch, Methods for fault tolerance in networks-on-chip, ACM Comput. Surv. (CSUR) 46 (1) (Oct. 2013) 1–32, article 8.
[10] A. Achballah, S. Othman, S. Saoud, Problems and challenges of emerging technology networks-on-chip: a review, Microprocess. Microsyst. 53 (Aug. 2017) 1–20.
[11] J. Wang, L. Huang, Q. Li, G. Li, A. Jantsch, Optimizing the location of ECC protection in network-on-chip, in: International Conference on Hardware/Software Codesign and System Synthesis (CODES), 2016, pp. 1–10.
[12] T. Majumber, M. Suri, V. Shekhar, NoC Router using STT-MRAM based hybrid buffers with error correction and limited flit retransmission, in: IEEE international Symposium on Circuits and Systems (ISCAS), May. 2015, pp. 2305–2308.
[13] F. Silva, W. Magalhães, J. Silveira, J. Ferreira, P. Magalhães, O. Lima Jr., C. Marcon, Evaluation of multiple bit upset tolerant codes for NoCs buffering, in: IEEE Latin American Symposium on Circuits & Systems (LASCAS), 2017, pp. 1–4.
[14] A. Pinheiro, J. Silveira, D. Tavares, F. Silva, C. Marcon, Optimized fault-tolerant buffer design for network-on-chip applications, in: IEEE 10th Latin American Symposium on Circuits & Systems (LASCAS), 2019, pp. 217–220.
[15] J. Gracia-Morán, L. Saiz-Adalid, D. Gil-Tomás, P. Gil-Vicente, Improving error correction codes for multiple-cell upsets in space applications, IEEE Trans. Very Large-Scale Integr. (TVLSI) 26 (10) (Oct. 2018) 2132–2142.
[16] J. Silveira, C. Marcon, P. Cortez, G. Barroso, J. Ferreira, R. Mota, Scenarios preprocessing approach for reconfiguration of fault-tolerant NoC based MPSoCs, Microprocess. Microsyst. 40 (Feb. 2016) 137–153.
[17] R. Hamming, Error detecting and error correcting codes, Bell Syst. Tech. J. 29 (2) (Apr. 1950) 147–160.
[18] S. Baeg, S. Wen, R. Wong, SRAM interleaving distance selection with a soft error failure model, IEEE Trans. Nucl. Sci. (TNS) 56 (4) (Aug. 2009) 2111–2118.
[19] E. Fujiwara, Code Design for Dependable Systems: Theory and Practical Applications, Wiley-Interscience, 2006, p. 702.
[20] C. Ogden, M. Mascagni, The impact of soft error event topography on the reliability of computer memories, IEEE Trans. Reliab. (TR) 66 (4) (Dec. 2017) 966–979.
[21] W. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers Inc., 2004.