IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 3 February 2019; revised 1 October 2019; accepted 8 October 2019.
Date of publication 17 October 2019; date of current version 6 December 2021.

Digital Object Identifier 10.1109/TETC.2019.2947617

Reduced Precision Redundancy for Reliable Processing of Data

SHANSHAN LIU, (Member, IEEE), KE CHEN, (Member, IEEE), PEDRO REVIRIEGO, (Senior Member, IEEE), WEIQIANG LIU, (Senior Member, IEEE), AHMED LOURI, (Fellow, IEEE), AND FABRIZIO LOMBARDI, (Fellow, IEEE)

S. Liu, K. Chen and F. Lombardi are with the Dept. of ECE, Northeastern University, Boston, MA 02115, USA
P. Reviriego is with Universidad Carlos III de Madrid, Av. Universidad 30, Leganes, Madrid, Spain
W. Liu is with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu 210016, China
A. Louri is with the Department of Electrical and Computer Engineering, George Washington University, Washington, DC 20052, USA
CORRESPONDING AUTHOR: F. LOMBARDI (lombardi@ece.neu.edu)

ABSTRACT Information is an integral part of the correct and reliable operation of today's computing systems. Data either stored or provided as input to computation processing modules must be tolerant to many externally and internally induced destructive phenomena such as soft errors and faults, often of a transient nature but also in large numbers, thus causing catastrophic system failures. Together with error tolerance, reliable operation must be provided by reducing the large overheads often encountered at system-level when employing redundancy. While information-based techniques can also be used in some of these schemes, the complexity and limited capabilities for implementing high order correction functions for decoding limit their application due to poor performance; therefore, *N* Modular Redundancy (*N*MR) is often employed. In *N*MR the correct output is given by majority voting among the *N* input copies of data. Reduced Precision Redundancy (RPR) has been advocated to reduce the redundancy, mostly for the case of $N = 3$; in a 3RPR scheme, one full precision (FP) input is needed while two inputs require reduced precision (RP) (usually by truncating some of the least significant bits (LSBs) in the input data). However, its decision logic is more complex than a 3MR scheme. This paper proposes a novel *N*RPR scheme with a simple comparison-based approach; the realistic case of $N = 5$ is considered as an example to explain in detail such proposed scheme; different arrangements for the redundancy (with three or four FP data copies) are considered. In addition to the design of the decision circuit, a probabilistic analysis is also pursued to determine the conditions by which RPR data is provided as output; it is shown that its probability is very small. Different applications of the proposed *N*RPR system are presented; in these applications, data is used either as memory output and/or for computing the discrete cosine transform. In both cases, the proposed 5RPR scheme shows considerable advantages in terms of redundancy management and reliable image processing.

INDEX TERMS Reduced precision redundancy, fault/error tolerance, memory

I. INTRODUCTION

Information (as binary data, either communicated or stored) plays a significant role in modern computer systems; today's applications require the acquisition of large amount of data (such as encountered in the so-called "Big Data" paradigm) as well as its processing for efficient computation [1]. Massive storage is employed to store large amount of data [2] such that it can be efficiently retrieved and subsequently processed by computational units in diverse application fields, such as data mining, machine learning and image processing.

Across the entire computation/memory hierarchy, correctness of data must be preserved to ensure that the entire process generates meaningful outcomes. Data correctness can be jeopardized by different causes, such as corruption, soft errors, faults as well as catastrophic failures [3].

Consider for example memories. Once operational, memories can be affected by soft errors; these errors should be taken into account when designing reliable chips [4]. There are many phenomena that can affect the correctness of data stored in memories. For example, radiation effects can cause

errors in memory cells as Single Event Upsets (SEUs) or functional failures in the control circuitry by Single Event Functional Interrupts (SEFIs) [5], [6]. These errors/failures may occur on a single or multiple bits, and affecting single or multiple words; in some cases, the entire memory can be affected as a catastrophic failure may result. Hardened techniques, in which redundancy is added to the unprotected module have been widely studied to deal with errors/failures [7], [8]. Reliability analysis and some trade-offs between reliability and protection cost are also studied in [9–11].

A class of hardened techniques applicable to memory relies on information-based redundancy. In this scheme, Error Correction Codes (ECCs) are used to detect and correct multiple bit errors on memory words [12], [13]; however they are not effective when t (the number of bits in error) in a given module is large. Redundant memory cells are added to each word of the original memory to store parity bits. Additional circuits are also required to implement decoding as needed by different classes of ECCs. Therefore, the hardware overhead includes the additional memory array and logic circuits, and depends on the number of parity bits and the complexity of the decoding process. A common class of ECCs is Hamming codes; these codes can correct single bit errors [14]. However they are unable to provide protection for memories in the case of multiple errors (random or correlated). ECCs with a higher error correction capability usually incur in complex implementations and often in longer latencies, so not always applicable to high performance systems. There are some ECCs that can correct multiple bit errors fast, such as Orthogonal Latin Square (OLS) codes [15–17]. These codes rely on one-step majority logic schemes; however for codes when dealing with a large number of errors, hardware overhead will be prohibitively large.

Another class of hardened techniques that can handle erroneous data in either multiple bit errors or functional failures (regardless of the number of bits affected) is *N* modular redundancy (*N*MR) [18]. In this scheme, the unprotected module (such as a single memory) is replicated *N* times (*N* is generally known as the order of replication); as criterion, majority voting among the *N* copies establishes the correctness of the data as final outcome. The *N*MR scheme can guarantee a correct output under simultaneous multiple faults (e.g., 3MR (i.e., $N = 3$) can handle one fault module and 5MR (i.e., $N = 5$) can handle two) but it incurs in a large overhead due to replication. An improved 3MR scheme is presented in [19] however error tolerance is limited to a subset of specific input signals in the system. Another scheme that is similar to 3MR but uses approximate voting, is presented in [20]. For three modules with *n*-bit data, only the upper *n-k* bits are checked in a pairwise fashion. If the absolute value of the subtraction between each pair is not bigger than "1" (which means that only the $k + 1$ lower bits may be affected by the error), all of the three copies are considered as "correct" and any of them will be provided as output. Otherwise, a "correct" pair (if there is at least one) of copies are checked by another decision logic; the $n - k$ upper bits of any of these two copies are provided as output. Padding with higher precision k bits that correspond to the average value of the k lower bits in the two copies is also utilized. Compared to 3MR, the scheme in [20] incurs in a lower area and power dissipation, but it requires a longer delay; it can only detect errors in one module or limited to the $k + 1$ lower bits in multiple modules and guarantee an output with a higher precision (but still inexact). Under multiple errors affecting the $n - k - 1$ upper bits in multiple modules, the scheme would fail. The information on the distribution of expected values and errors at the voter inputs can also be used to design *N*MR schemes that have the ability to correct more errors. For example, the soft *N*MR scheme proposed in [21] considers the input distributions to the voters and formulates voting as a detection problem using communication theory techniques to obtain the result that is more likely to be correct. Finally, techniques at different abstraction layers can be combined into cross-layer schemes that can further improve fault tolerance for some applications like image processing [22].

In some applications, such as signal and image processing, a limited range of deviation from the correct result can be tolerated. For such applications, Reduced Precision Redundancy (RPR) that can handle one erroneous module has been proposed as alternative [23]. In this scheme, the difference between the full precision (FP) and the two reduced precision (RP) modules is calculated and compared with a threshold error value; if the subtraction result is smaller than the threshold, this means that the results are acceptable; so based on the decision logic circuit, data from the FP module can be provided as output. Else, the outputs of the two RP modules are further analyzed to determine the single faulty one. This scheme can significantly reduce the overhead due to redundancy because two copies are implemented by reduced precision. It should also be noted that the decision circuit (subtraction and comparison logic) is significantly more complex than the voting logic used for the 3MR with full precision. This scheme has been analyzed in [24] for applications in which addition is employed as processing step.

Recently, the RPR schemes of [24] and [25] have been proposed as improvements for adders and Multiplication and Accumulation (MAC) computation in signed integer format to reduce the hardware overhead in the decision logic as well as the error due to the reduced precision. However, both these schemes are based on specific features of the implemented arithmetic computation modules; moreover, as the order of replication is three, they can tolerate only a single faulty module.

In applications in which data correctness is affected from multiple simultaneous sources (of either permanent or transient nature or long mission time with limited corrective action), a higher order of redundancy is often needed; the error rate for the data is significantly larger than for example the fault rate in logic circuits for arithmetic computation. At $N = 5$ as an example of the next level of redundancy, which is the most realistic case; for a larger value of *N,* the design is more complex and accordingly the hardware overhead is very large [26], RPR has not been analyzed even though

intuitively it could yield significant improvements for many applications. RPR has been considered in [25] with respect to arithmetic computation and the format of the operands. The binary nature of data (such as applicable to memories) also necessitates a different technique to RPR management, especially when the error rate increases and two modules may be faulty generating incorrect values. In this case, ECCs require a large number of parity bits and a decoder with high complexity to provide a strong error correction capability, thus introducing a large memory overhead and decoding latency; moreover, *N*MR would incur in a large hardware overhead due to the N replication order for full precision. Therefore, an RPR scheme with a higher order of redundancy that can efficiently reduce both the overhead of redundant modules and logic complexity is needed to deal with multiple module failures.

In this paper, a novel *N*RPR scheme is proposed as applicable to tolerance and processing of data; this scheme is based on a comparison-based approach for *N*RPR to ensure that the decision logic does not account for a significant design overhead. The realistic case of $N = 5$ is considered as an example and different arrangements for the redundancy (with three or four FP data copies) are considered. In addition to the design of the decision circuit, a probabilistic analysis is also pursued to determine the conditions by which RPR data is provided as output; it is shown that the probability is very small. Different applications of the proposed *N*RPR system are presented; in these applications, data is used either as memory output and/or for computing the discrete cosine transform. In both cases, the 5RPR scheme is proposed as an example and shows considerable advantages in terms of redundancy management and reliable image processing.

The rest of the paper is organized as follows. Section II deals with a brief review of existing hardware redundancy techniques, including *N*MR and RPR schemes (for this last case the current schemes of [23] with a single FP copy for $N = 3$ are discussed). Section III presents the proposed scheme with emphasis on syndrome generation and checking. Section IV presents a probabilistic analysis for establishing the probability that RP data is delivered at the scheme output. Section V evaluates and compares the proposed schemes with other modular redundant schemes found in the technical literature; evaluation is pursued with respect to design of the decision logic circuitry under different data size. Section VI presents two cases for the application of the proposed 5RPR scheme, namely memory and Discrete Cosine Transform. Finally, the paper ends with the conclusions in Section VII.

II. PRELIMINARIES

This section provides a brief review of relevant material for redundant design techniques found in the literature [18], [23].

A. N MODULAR REDUNDANCY

N Modular Redundancy (*N*MR) is widely used for fault/erroneous tolerance at system level. *N*MR is based on voting among N copies of the original module; therefore $N = 2t + 1$ modules are required to guarantee a correct output under a number of simultaneous faulty modules not exceeding t [18]. For *N*MR system the majority of the modules ($t + 1$ at least) is always assumed to be operating correctly, hence *N*MR masks all erroneous values in the minority. The *N*MR scheme is shown in Figure 1 in block diagram form; all C_N^{t+1} combinations for the N modules must be considered when designing the majority voting circuit. The output of the voting circuitry is always correct provided the number of erroneous modules does not exceed t.

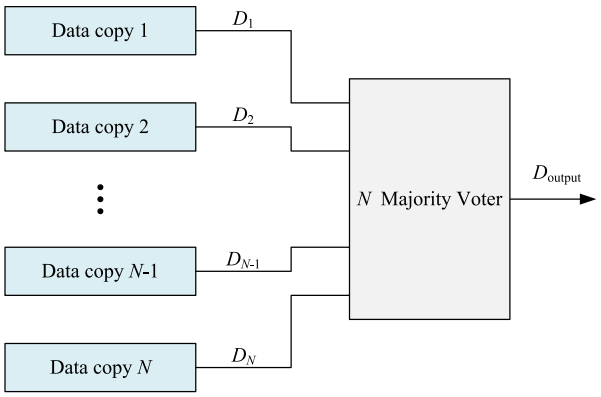

FIGURE 1. *N*MR scheme.

B. REDUCED PRECISION REDUNDANCY

The so-called Reduced Precision Redundancy (RPR) scheme can be an alternative in some applications such as image processing. For example, [23] and [25] have proposed RPR schemes in which the order of replication is 3; 2 modules compute on a reduced precision basis, i.e., only a module has full precision (Figure 2).

The design of a reduced precision module lends itself to approximate computing because for example in an arithmetic circuit, the least significant bits (LSBs) are truncated in each of the reduced precision modules. RPR uses a full precision copy of the circuit and two reduced precision copies by implementing a decision logic to correct errors. The difference between the full precision (FP) and RP copies needs to

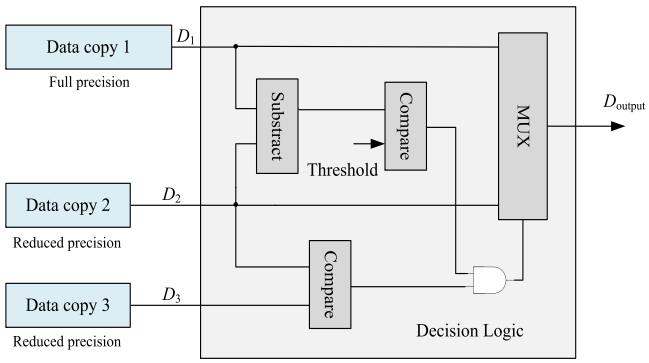

FIGURE 2. RPR scheme with one FP copy and two RP copies ($N = 3$).

be computed and compared to a threshold. Therefore, a subtraction and comparison are needed. The decision logic for the RPR scheme is significantly more complex than the majority voting logic used for a FP 3MR; however, the RPR scheme significantly lowers the total hardware overhead because the RP copies are usually significantly smaller than the FP implementation and the processing module is significantly more complex than the decision logic.

A disadvantage of this scheme is that small errors that only affect the LSBs cannot be corrected; this is an inherent problem of RPR schemes. There are two cases on which small errors may occur. One case is that small errors on the LSBs of the FP data copy cause a smaller subtraction between the FP and RP copies than the specified threshold; thus those errors cannot be detected and a FP with small errors will be used as the output. The other case is that when RP data is used as the output, the truncated LSBs cannot be recovered so that small errors occur. This however is not a problem for many applications in which either most failures are due to large errors that affect some upper bits of the data and cause for example a significant deviation from the correct result, or the output is inherently error-tolerant (such as for image processing). To reduce the error in output data and improve performance (such as the delay in the decision logic), [25] has exploited complementary features in signed operation for MAC. Different from the RPR scheme of [23] that has a single threshold value, the threshold value in the scheme of [25] is reduced by half of the default value when the two operands have different signs. This design selection is based on the observation that errors will compensate when the operands have different signs. By utilizing a smaller threshold, the rate of reduced accuracy in the output is reduced, i.e., system precision improves. However, the scheme of [25] is applicable only when computation in signed integer format is executed for processing; it is not applicable to protect data from errors because no computation is involved in this process.

Similar to arithmetic circuits, data can be generated and stored in a RP memory; for data as a long word in integer format (hence with a very large value) its LSBs are truncated such that a small portion of its value is lost at the expense of a reduced capacity of the memory array. This feature will be utilized in a subsequent section when the proposed scheme is applied to memory systems.

III. PROPOSED SCHEME

In this section, a new RPR scheme (denoted as *N*RPR) is proposed. This design is applicable to data replication to establish correctness in output once for example data is read from different storage units (such as memories). In general, an *N*RPR scheme needs the same number of copies as the *N*MR scheme but several copies are of RP, i.e., $2t + 1$ modules are required in case of t erroneous modules. In this paper, we take the realistic case of $N = 5$ as an example to illustrate the proposed scheme, i.e., the 5RPR. In this case, data can then be represented in FP or RP; data can be also correct (not erroneous) or incorrect (erroneous) with respect to its initial precision type.

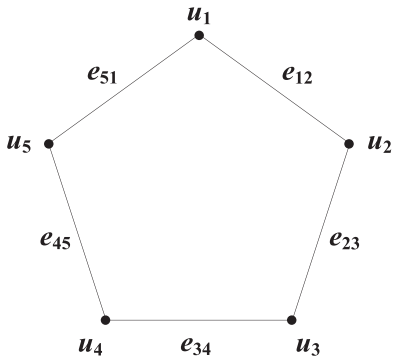

FIGURE 3. Connection assignment graph of 5RPR scheme.

The proposed scheme is based on a comparison-based technique as initially detailed in [27] for system-level diagnosis and its graph theoretical approach [28]. Recall that a diagnosable system made of N units (in this case the N copies of the data to be protected) can be represented by a set $\boldsymbol{U} = \{u_1, u_2, \ldots, u_N\}$. A connection assignment refers to the complete collection of comparisons of each pair of two units, this assignment is represented by an undirected graph $\boldsymbol{G} = (\boldsymbol{U}, \boldsymbol{E})$ (such as shown in Figure 3). In Figure 3, each vertex represents a single unit $u_i \in \boldsymbol{U}\,(i = 1, 2, \ldots, 5)$ and each edge e_{ij} in $\boldsymbol{E}$ is labelled by the so-called syndrome as weight denoting the outcome of the comparison between the two connected vertices (i.e., 0 if they are equal, 1 otherwise). According to all syndromes for the outcomes of the comparisons in the connection assignment, it is possible to prove the correctness of at least one data copy (corresponding to a vertex in the graph). The comparison-based approach in [27], [28] is extended in our proposed *N*RPR scheme with some RP copies to determine a correct copy (preferably with FP or alternatively RP) at a lower cost.

The proposed *N*RPR scheme is illustrated for $N = 5$ (i.e., 5RPR) as an example. Figure 3 shows the undirected graph for 5 units; vertices u_1, u_2, u_3, u_4 and u_5 represent the replicated data copies. Each edge represents the comparison on a bit by bit basis between the two connected vertices (this operation can be simply implemented by using *xor* gates). Similar to the method of [27], [28], the comparison results are given by the syndromes S_i and defined as follows:

$$S_1 = D_1 \; xor \; D_2 \tag{1}$$

$$S_2 = D_2 \; xor \; D_3 \tag{2}$$

$$S_3 = D_3 \; xor \; D_4 \tag{3}$$

$$S_4 = D_4 \; xor \; D_5 \tag{4}$$

$$S_5 = D_5 \; xor \; D_1, \tag{5}$$

where D_i is the i^{th} data copy (for example stored in the i^{th} memory). The value of each syndrome bit can be 0 or 1; note that a single syndrome with value 0 does not imply data correctness because both data copies in a vertex pair can be

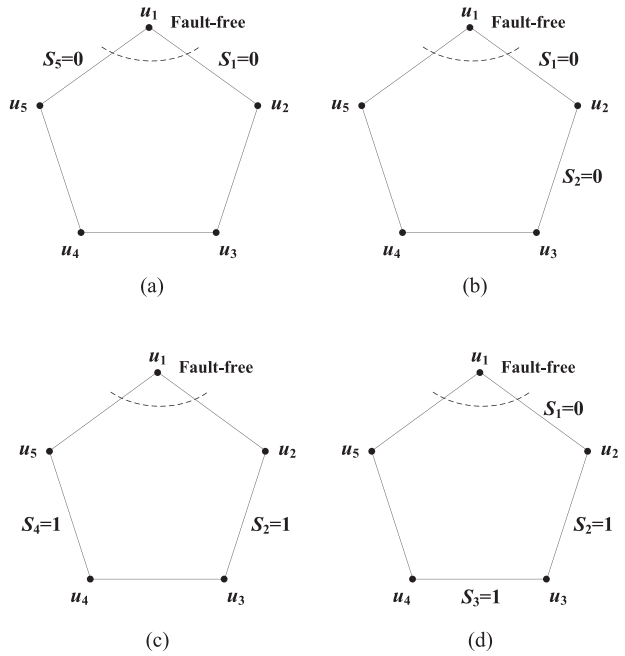

FIGURE 4. Examples of relationships between syndromes and the correctness of data as per observation 1 to 3: (a) $S_1 = 0$ **and** $S_5 = 0$**, (b)** $S_1 = 0$ **and** $S_2 = 0$**, (c)** $S_2 = 1$ **and** $S_4 = 1$**, (d)** $S_1 = 0, S_2 = 1$ **and** $S_3 = 1$**.**

equally incorrect. Different from [27], [28], in the proposed scheme comparison between RP and FP copies occurs only for the common part (i.e., the LSBs that are truncated in the RP copies are not compared). In this respect, we present the following observations (and related proofs) to validate the proposed scheme.

Observation 1: A data copy (corresponding to u_i) is correct (so not erroneous) if

$$\begin{cases} S_i = 0, & S_5 = 0 \quad \text{if } i = 1 \\ S_i = 0, & S_{i-1} = 0 \quad \text{if } 1 < i \leq 5, \end{cases} \tag{6}$$

or

$$\begin{cases} S_i = 0, & S_{i+1} = 0 \quad \text{if } 1 \leq i < 5 \\ S_i = 0, & S_1 = 0 \quad \text{if } i = 5, \end{cases} \tag{7}$$

Proof: A 0 syndrome bit indicates that both of the two compared copies of data are either correct or incorrect. If the two syndrome bits of a vertex are 0, then the three corresponding data copies should have the same status, i.e., all correct or all incorrect. However, the second scenario is not possible because by assumption t cannot exceed 2. □

Figure 4 (a) and (b) show two equivalent scenarios as examples: when $S_1 = 0$ and $S_5 = 0$, or $S_1 = 0$ and $S_2 = 0$, u_1 must be correct.

Observation 2: A data copy (corresponding to u_i) is correct (so not erroneous) if

$$\begin{cases} S_{i+1} = 1, & S_{i+3} = 1 \quad \text{if } 1 \leq i \leq 2 \\ S_{i-2} = 1, & S_{i+1} = 1 \quad \text{if } 2 < i < 5 \\ S_{i-2} = 1, & S_{i-4} = 1 \quad \text{if } i = 5. \end{cases} \tag{8}$$

Proof: A non-zero syndrome bit indicates that at least one of the two compared vertices is erroneous. If two discontinuous syndrome bits are equal to one, there must be one erroneous data copy in each pair of compared vertices, so two erroneous copies in total. Therefore, this proves that the other copies (vertices) are correct as there are no more than two erroneous copies in a 5RPR by assumption. Figure 4 (c) is an example; if $S_2 = 1$ and $S_4 = 1$, u_1 must be correct. □

Observation 3: A data copy is correct if the string given by three continuous syndrome bits (that started from such vertex) is equal to "011", i.e., u_i is correct if:

$$\begin{cases} S_i = 0, & S_{i+1} = 1, & S_{i+2} = 1 \quad \text{if } 1 \leq i \leq 3 \\ S_i = 0, & S_{i+1} = 1, & S_1 = 1 \quad \text{if } i = 4 \\ S_i = 0, & S_1 = 1, & S_2 = 1 \quad \text{if } i = 5. \end{cases} \tag{9}$$

Proof: A string with three continuous syndromes of value "011" corresponds to the cases in which one or two incorrect copies have occurred. If there is just one incorrect data, then it is identified by the copy (vertex) between the two "1" syndrome bits. If there are two incorrect copies, they corresponds to the last two vertices in the path identified by the string. Therefore, the copy that is the starting point the path should be correct. Figure 4(d) shows an example: if $S_1 = 0, S_2 = 1$, and $S_3 = 1$, u_1 must be correct. □

Based on the above observations, only few comparisons on the syndrome bits are sufficient to generate the correct and FP/RP output using the proposed comparison-based approach. The algorithm used to implement the proposed 5RPR scheme is given as follows:

Input: 5 copies of the same data (for example a word from each memory, 5 independent memory units); q copies are FP, 5-q copies are of equal RP (for RP k-bit LSBs are truncated from each of the 5-q data copies). The FP data copies correspond to the first vertices, u_1 to u_q. When $q = 3$, the algorithm is given as:

1) Compare each pair of adjacent copies to calculate the syndrome bits as per equations (1) to (5).
2) If all syndrome bits are zero, then there is no erroneous copy, D_1 is provided as output and terminate the algorithm. If not, go to the next step.
3) If $S_1 = 0$ and $S_2 = 0$, or $S_2 = 1$ and $S_4 = 1$, or $S_1 = 0, S_2 = 1$, and $S_3 = 1, D_1$ is correct and provided as output.
4) If $S_2 = 0$ and $S_3 = 0, D_2$ is correct and provided as output.
5) If $S_1 = 1$ and $S_4 = 1, D_3$ is determined to be correct and provided as output.
6) In all other cases, the RP data D_5 (correct) padded with the LSBs of D_1 is provided as output. In this case, small errors may happen at the LSBs and D_1 may be affected by errors, e.g., the data pattern of "11000" with errors on LSBs of the first copy.

The number of RP copies cannot be in the majority; when q is either 1 or 2, an all-zero syndrome can be erroneously generated for $N = 5$. For example, when 2 FP vertices (u_1

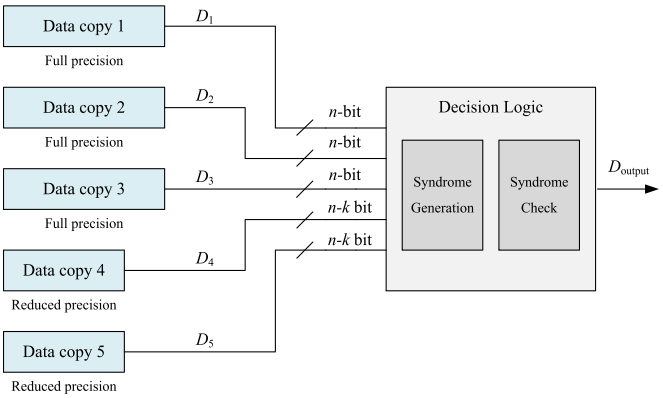

FIGURE 5. Proposed 5RPR scheme with three FP copies.

and u_2) are used, an all zero syndrome can be obtained based on equations (1) to (5) for same errors on the LSBs of D_1 and D_2. Then the decision logic will consider this as correct and erroneous data will be provided to the output. The 5RPR scheme with three FP copies is shown in Figure 5; the syndrome generator implements step 1), while the syndrome check is used for steps 2) to 6).

The proposed approach is therefore based on comparisons; while 5RPR is the most realistic scheme, the proposed algorithm can be also extended to other RPR scheme of higher order of redundancy without loss of generality. In this case, as the number of modules increases, more equations are needed to generate the syndrome bits by comparing each pair of connected vertices in the graph. The logic for checking the syndrome must also be extended to account for a larger number of combinations than for 5RPR. In any case, the logic circuitry is expected to be simpler than for existing RPR solutions requiring arithmetic operations when determining the correct module.

IV. PROBABILISTIC ANALYSIS

As some LSBs are not presented in the RP copies, these positions are not compared with the entire FP data copies, because comparisons only focus on the upper bits (e.g., when only u_4 and u_5 are RP, the LSBs of D_3 and D_1 (that are FP) cannot be checked by Eqs. (3) and (5)). Therefore, the corresponding syndrome bits cannot be determined and the value may be 0 or 1. In this case, the syndromes for different patterns may be the same, so they cannot be distinguished. This will result in a situation in which no FP data copy can be assessed as correct while a RP copy can be proved to be correct. Therefore, in some cases RP data is provided at the output. This is shown clearly in Table 1, in which the syndromes for all possible data patterns in a 5RPR scheme with three FP data copies (u_1, u_2 and u_3) are presented. For the gray colored rows, the syndromes can in some cases be the same. For example, they can be "11000" in all four cases, so that we cannot determine which copy of FP data (i.e., D_1, D_2, and D_3) is correct. However, in this case the RP data D_5 is correct, so it can be provided as output and padding the LSBs is required causing an output to have inexact data

TABLE 1. Patterns & syndrome bits for 5RPR scheme with 3 FP copies.

Data Pattern	S_1	S_2	S_3	S_4	S_5	Output
10000	1	0	0	0	X	D_2
01000	**1**	**1**	**0**	**0**	**0**	$\boldsymbol{D_5}$
00100	**0**	**1**	$\boldsymbol{X}$	**0**	**0**	$\boldsymbol{D_1}$ **when** $\boldsymbol{S_3 = 1}$ $\boldsymbol{D_5}$ **when** $\boldsymbol{S_3 = 0}$
00010	0	0	1	1	0	D_1
00001	0	0	0	1	1	D_1
11000	$\boldsymbol{X}$	**1**	**0**	**0**	$\boldsymbol{X}$	$\boldsymbol{D_5}$
10100	**1**	**1**	$\boldsymbol{X}$	**0**	$\boldsymbol{X}$	$\boldsymbol{D_5}$
10010	1	0	1	1	X	D_3
10001	1	0	0	1	X	D_2
01100	**1**	$\boldsymbol{X}$	$\boldsymbol{X}$	**0**	**0**	$\boldsymbol{D_5}$
01010	1	1	1	1	0	D_1
01001	1	1	0	1	1	D_1
00110	0	1	X	1	0	D_1
00101	0	1	X	1	1	D_1
00011	0	0	1	X	1	D_1

1 (0) on the i^{th} position of the data pattern denotes that the i^{th} copy is incorrect (correct) $(1 \le i \le 5)$.
X *denotes that the value can be 0 or 1.*

(inexact data here refers to data that has the correct upper bits but may have errors on the padding for the LSBs). Therefore this can be tolerated only for some applications as discussed previously.

The probability of outputting RP data (*i.e.*, $P_{inexact\_5RPR3}$) is analyzed next; models are established based on the assumption that errors are independent to permit a simple evaluation. Further details on the derivations of the equations are provided in an Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TETC.2019.2947617.

By denoting the probability of an incorrect data copy as p, the probability of no more than two erroneous data copies $P_{\le 2\_units}$ (as considered in the proposed 5RPR scheme) is given by:

$$P_{\le 2\_units} = (1-p)^5 + C_5^1 p(1-p)^4 + C_5^2 p^2 (1-p)^3, \quad (10)$$

where C_5^1 (C_5^2) is the number of combinations for one (two) incorrect copy (ies) and four (three) correct copies on the five data copies.

Then we can have the probability $P_{i=j}$ for the case of the same erroneous status on the i^{th} and j^{th} copies $(1 \le i \le 5, 1 \le j \le 5)$, and the probability P_{i_LSBs} for the case of only an erroneous status on the i^{th} copy LSBs. They are given by Eq. (11) and (12) respectively.

$$P_{i=j} = (1-p)^3 \cdot \sum_{l=1}^{n} C_n^l \left[{p_{bit}}^l \cdot (1-p_{bit})^{n-l} \right]^2 \quad (11)$$

$$P_{i\_LSBs} = (1-p)^4 \cdot (1-p_{bit})^{n-k} \cdot \sum_{l=1}^{k} C_k^l p_{bit}^l (1-p_{bit})^{k-l}, \quad (12)$$

where p_{bit} is the probability of an erroneous bit, C_n^l is the number of combinations for l erroneous bits and $n-l$ correct bits on the n-bit incorrect data copy, and C_k^l is the number of combinations for l erroneous bits and $k-l$ correct bits on the

TABLE 2. Patterns & syndrome bits for 5RPR scheme with 4 FP copies.

Data Pattern	S_1	S_2	S_3	S_4	S_5	Output
10000	1	0	0	0	X	D_2
01000	1	1	0	0	0	D_3
00100	0	1	1	0	0	D_1
00010	0	0	1	X	0	D_1
00001	0	0	0	1	1	D_1
11000	X	**1**	**0**	**0**	X	D_5 **when** $S_1 = 0$
						D_3 **when** $S_1 = 1$
10100	**1**	**1**	**1**	**0**	X	D_5
10010	**1**	**0**	**1**	X	X	D_5
10001	1	0	0	1	X	D_2
01100	**1**	X	**1**	**0**	**0**	D_5
01010	**1**	**1**	**1**	X	**0**	D_5
01001	1	1	0	1	1	D_1
00110	**0**	**1**	X	X	**0**	D_5 **when** $S_3 = 0$
						D_1 **when** $S_3 = 1$
00101	0	1	1	1	1	D_1
00011	0	0	1	X	1	D_1

1 (0) on the ith position of the data pattern denotes that the i^{th} copy is incorrect (correct) ($1 \le i \le 5$).
X means the value can be 0 or 1.

k-bit LSBs of the incorrect data copy. The term $(1-p)^3$ in Eq. (11) is the probability of the other three data copies being correct as at most two copies can be erroneous. Similarly, the term $(1-p)^4$ in Eq. (12) is the probability of the other four data copies being correct. Consider data to be made of a n-bit word; as the probability of one n-bit word error is p, the probability of one correct word is $1-p$; so the probability of a correct (error free) bit is $\sqrt[n]{1-p}$ because all events are supposed to be independent. So p_{bit} is given by

$$p_{bit} = 1 - \sqrt[n]{1-p}. \tag{13}$$

As shown in Table 1, RP data (D_5) is output only when the data pattern is "01000", "00100" with errors on the truncated LSBs that cause $S_3 = 0, "11000", "10100", \text{or} "01100"$. Therefore, as per Eqs. (10), (12) and (13), we can find $P_{inexact\_5RPR3}$ for the 5RPR scheme with three FP data copies. This is given by Eq. (14), where P_i is the probability of only the i^{th} copy to be incorrect (i.e., $P_i = p(1-p)^4$), and $P_{i,j}$ is the probability of both copies i^{th} and j^{th} being incorrect (i.e., $P_{i,j} = p^2(1-p)^3$).

$$\begin{aligned} P_{inexact\_5RPR3} &= \frac{P_2 + P_{3\_LSBs} + P_{1,2} + P_{1,3} + P_{2,3}}{P_{\le 2\_units}} \\ &= \frac{p(1-p)^4 + (1-p)^4 \cdot (1-p_{bit})^{n-k} \cdot \sum_{l=1}^{k} C_k^l \cdot {p_{bit}}^l \cdot (1-p_{bit})^{k-l} + 3p^2(1-p)^3}{(1-p)^5 + C_5^1 p(1-p)^4 + C_5^2 p^2 (1-p)^3} \\ &= \frac{(1-p) \cdot (p + (\sqrt[n]{1-p})^{n-k} \cdot \sum_{l=1}^{k} C_k^l \cdot (1-\sqrt[n]{1-p})^l \cdot (\sqrt[n]{1-p})^{k-l}) + 3p^2}{(1-p)^2 + 5p(1-p) + 10p^2}. \end{aligned} \tag{14}$$

For a 5RPR scheme with four FP data copies $(u_1, u_2, u_3$ and $u_4)$, the syndromes for all possible patterns are given in Table 2. As per Eqs. (10), (11) and (13),

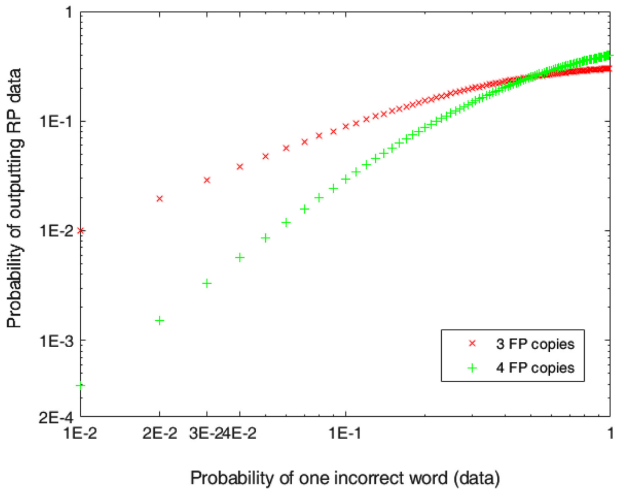

FIGURE 6. Probability of outputting RP data for the proposed 5RPR schemes.

the probability of outputting RP data $P_{inexact\_5RPR4}$ can be calculated in a similar manner as Eq. (14). Eq. (15) is then obtained.

$$\begin{aligned} P_{inexact\_5RPR4} &= \frac{P_{1=2} + P_{1,3} + P_{1,4} + P_{2,3} + P_{2,4} + P_{3=4}}{P_{\le 2\_units}} \\ &= \frac{2(1-p)^3 \cdot \sum_{l=1}^{n} C_n^l \left[{p_{bit}}^l \cdot (1-p_{bit})^{n-l}\right]^2 + 4p^2(1-p)^3}{(1-p)^5 + C_5^1 p(1-p)^4 + C_5^2 p^2 (1-p)^3} \\ &= \frac{2 \cdot \sum_{l=1}^{n} C_n^l \cdot (1-\sqrt[n]{1-p})^{2l} \cdot (\sqrt[n]{1-p})^{2(n-l)} + 4p^2}{(1-p)^2 + 5p(1-p) + 10p^2}, \end{aligned} \tag{15}$$

Figure 6 plots both probabilities for 32-bit word data by truncating 2-bit LSBs in the RP data (i.e., $n = 32, k = 2$) as example. The probability of outputting RP data is low at a small probability of an incorrect word. For example, the 5RPR scheme with three FP data copies has a 9.22 percent probability of outputting RP data; this probability for a 5RPR with four FP data copies is only 2.98 percent at a probability of an incorrect single word of 10 percent. Although the probability of outputting RP data will increase, in the case of a higher probability of an incorrect word it is also likely that two incorrect data copies will occur in the 5RPR.

For comparative purposes, a similar analysis is pursued for the RPR scheme proposed in [23] so that the probability of outputting RP data can be calculated. Assume that erroneous behavior is independent. Let the probability of an incorrect data copy be denoted by p, so the probability of no more than an incorrect copy $P_{\le 1\_unit}$ is given by:

$$P_{\le 1\_units} = (1-p)^3 + C_3^1 p(1-p)^2, \tag{16}$$

where C_3^1 is the number of combinations for one incorrect copy and two correct copies on the three data copies.

The probability of outputting RP data $P_{inexact\_3RPR}$ is obtained from Eq. (17); so for example the result for applying the scheme of [23] for a 32-bit data by truncating 2-bit

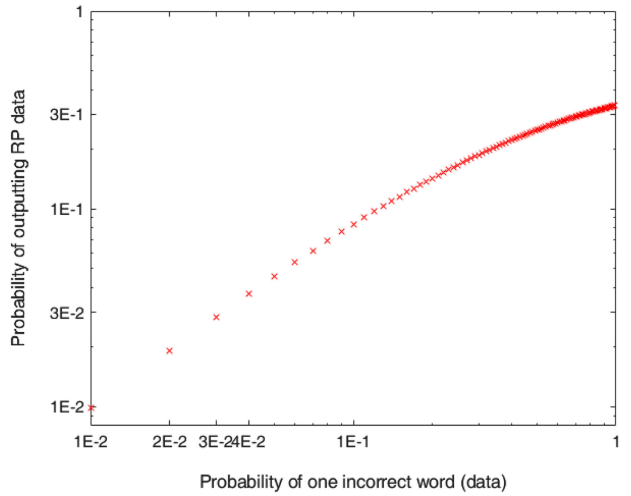

FIGURE 7. Probability of outputting RP data for the 3RPR scheme of [23].

LSBs is plotted in Figure 7. $P_{inexact\_3RPR}$ is 8.33 percent at a probability of an incorrect single word of 10 percent. At a small probability of a word to be incorrect, the probability of outputting RP data is extremely low.

$$P_{inexact\_3RPR} = \frac{p(1-p)^2}{(1-p)^3 + C_3^1 p(1-p)^2} = \frac{p}{1+2p}. \quad (17)$$

These results show that the scheme of [23] has a similar probability of outputting RP data as the 5RPR scheme with three FP copies, thus it is best suited when a single erroneous data copy is present (so occurring at a low error probability in data). If two erroneous copies of data occur (due to accumulation of errors over a long mission time for example), the scheme of [23] as well as a 3MR will yield an incorrect output. In these cases, 5 copies of data are required and the proposed 5RPR or the 5MR schemes should be used.

V. EVALUATION

The previous and proposed schemes have been implemented in HDL and mapped to a 65nm library from TSMC using Design Compiler, and place-and-route for a design has been implemented using Encounter. The results presented correspond to the final circuits obtained after place-and-route.

Data of different word width have been considered in the evaluation. The synthesis tool was set to reduce area, power consumption and delay in the decision circuits. Table 3 shows the results for the 5RPR scheme with three FP data copies and the 5MR scheme for data of 8/16/32/64-bit words.

5RPR schemes with different truncation levels in the RP data have also been evaluated. The decision circuit for the 5RPR scheme is better than for voting in the 5MR scheme in terms of area, power and delay in all cases. When 25 percent of the bits are truncated in the RP data for the 5RPR scheme, significant savings in area, delay and power are achieved; for example, 39.35 percent area, 21.65 percent delay, and 42.42 percent power can be saved in the case of data as a 32-bit word.

TABLE 3. Results for the decision circuit used by 5RPR scheme with three FP copies and 5MR scheme.

Scheme		Area(μm^2)	Delay(ns)	Power(mW)
8-bit	5RPR3_0	339.48	0.62	0.12
	5RPR3_2	309.19	0.60	0.11
	5MR	393.20	0.70	0.15
16-bit	5RPR3_0	619.52	0.75	0.20
	5RPR3_2	592.08	0.71	0.19
	5RPR3_4	559.51	0.68	0.18
	5MR	802.97	0.81	0.28
	(32,16) OLS	1419.06	0.78	0.57
32-bit	5RPR3_0	1395.06	0.80	0.51
	5RPR3_2	1355.05	0.79	0.50
	5RPR3_4	1243.61	0.78	0.43
	5RPR3_6	1210.46	0.77	0.39
	5RPR3_8	1175.03	0.76	0.38
	5MR	1937.42	0.97	0.66
64-bit	5RPR3_0	2727.25	0.91	0.95
	5RPR3_2	2687.24	0.90	0.93
	5RPR3_4	2644.38	0.88	0.91
	5RPR3_6	2610.66	0.88	0.90
	5RPR3_8	2574.65	0.87	0.89
	5RPR3_10	2538.08	0.86	0.88
	5RPR3_12	2506.07	0.85	0.87
	5RPR3_14	2469.50	0.84	0.85
	5RPR3_16	2436.35	0.83	0.84
	5MR	3833.69	1.13	1.12
	(96,64) OLS	5122.45	1.01	2.08

5RPR3_k (k = {0, 2, 4, . . . , 16}) denotes the 5RPR scheme in which k bits have been reduced in the two RP copies.

When using the 5RPR scheme by not truncating any bits (i.e., 5RPR3_0 in Table 3), the overhead is still lower than the 5MR scheme due to the proposed low complexity decision circuit. In the 5MR scheme, the majority voting circuit needs to check all C_5^3 combinations of the three correct data copies. 10 xor logic structures are therefore needed and each xor logic has $3 * n$ inputs (n is the data size). In the proposed 5RPR scheme, only five logic structures are used to generate the syndrome bits, and in each xor logic, two data copies are compared, i.e., $2 * n$ inputs. Several small logic blocks are used to determine the output (e.g., five small logic blocks in which there are just 4 inputs in four and 6 inputs in the last one are needed for the 5RPR scheme with three RP copies as discussed previously in Section III). Therefore, the simpler decision circuit in 5RPR outperforms the majority voting circuit in 5MR in terms of area, delay and power, and these advantageous features increase for a larger value of n. For example, in Table 3 28.86 percent area, 19.47 percent delay, and 15.18 percent power can be saved when protecting 64-bit data.

Table 4 shows the results for the 5RPR scheme with 4 FP data copies and a 5MR scheme for input data of 8/16/32/64-bit words. Also in this case, the 5RPR scheme reduces the decision logic circuit in all cases when compared to the 5MR

TABLE 4. Results for the decision circuit used by 5RPR scheme with four FP copies and 5MR scheme.

Scheme		Area(μm^2)	Delay(ns)	Power(mW)
8-bit	5RPR4_0	338.33	0.62	0.12
	5RPR4_2	317.19	0.60	0.11
	5MR	393.20	0.70	0.15
16-bit	5RPR4_0	617.80	0.75	0.20
	5RPR4_2	607.52	0.71	0.20
	5RPR4_4	589.23	0.69	0.19
	5MR	802.97	0.81	0.28
	(32,16) OLS	1419.06	0.78	0.57
32-bit	5RPR4_0	1392.77	0.81	0.51
	5RPR4_2	1365.34	0.80	0.50
	5RPR4_4	1279.61	0.79	0.43
	5RPR4_6	1261.89	0.78	0.42
	5RPR4_8	1241.32	0.77	0.42
	5MR	1937.42	0.97	0.66
64-bit	5RPR4_0	2727.82	0.92	0.95
	5RPR4_2	2701.53	0.91	0.94
	5RPR4_4	2680.95	0.90	0.93
	5RPR4_6	2658.67	0.89	0.91
	5RPR4_8	2630.66	0.89	0.90
	5RPR4_10	2607.80	0.87	0.89
	5RPR4_12	2579.23	0.86	0.88
	5RPR4_14	2564.37	0.86	0.88
	5RPR4_16	2547.79	0.84	0.87
	5MR	3833.69	1.13	1.12
	(96,64) OLS	5122.45	1.01	2.08

5RPR4_k (k = {0, 2, 4, . . . , 16}) denotes the 5RPR scheme in which k bits have been reduced in the one RP copy.

TABLE 5. Results for the decision circuit used by 3RPR [21], ITDMR [20] and 3MR schemes.

Scheme		Area(μm^2)	Delay(ns)	Power(mW)
8-bit	3RPR_2	434.92	0.89	0.16
	3MR	278.90	0.53	0.10
	ITDMR	267.47	0.54	0.09
16-bit	3RPR_2	961.28	1.09	0.32
	3RPR_4	891.56	1.07	0.29
	3MR	498.36	0.61	0.17
	ITDMR	477.86	0.66	0.16
32-bit	3RPR_2	2033.43	1.27	0.68
	3RPR_4	1925.99	1.26	0.66
	3RPR_6	1879.70	1.24	0.63
	3RPR_8	1783.11	1.23	0.60
	3MR	1213.32	0.72	0.42
	ITDMR	989.57	0.81	0.33
64-bit	3RPR_2	4758.97	1.63	2.68
	3RPR_4	4643.52	1.62	2.52
	3RPR_6	4517.79	1.59	2.31
	3RPR_8	4452.06	1.55	2.17
	3RPR_10	4338.91	1.53	2.00
	3RPR_12	4263.47	1.51	1.80
	3RPR_14	4201.17	1.49	1.77
	3RPR_16	4112.59	1.45	1.55
	3MR	2395.77	0.88	0.84
	ITDMR	2119.43	0.93	0.75

3RPR_k (k = {0, 2, 4, . . . , 16}) denotes the 3RPR scheme in which k bits have been reduced in the two RP copies.

scheme. As expected the saving in area and delay are marginally smaller compared to the case of a 5RPR scheme with 3 FP data copies.

For comparison, double error correction OLS (DEC OLS) codes have also been implemented and compared (this is evaluated for data sizes of 16 and 64-bit for which DEC OLS codes exist) to the proposed scheme. The results are given in Tables 3 and 4, in which the area/power correspond to both the encoder and the decoder, while the delay is for the decoder that is used to correct errors (so for the circuit that incurs in the largest delay). Compared to the 5RPR and 5MR schemes, DEC OLS codes have a larger overhead in terms of area and power, and incur in a moderate latency. Moreover, as ECCs parity bits are stored on each word, a module storing the (32, 16) OLS code has the same size as two copies of 16-bit data, and the (96, 64) OLS code has the size of one and a half copies. This leads to a much lower hardware overhead than the 5RPR and 5MR schemes. However, recall that ECCs focus on correcting several bit errors, while 5RPR and 5MR schemes focus on correcting the failure of an entire module regardless of the number of erroneous bits. In the presence of three bit errors per data or more, DEC OLS codes that can only correct double bit errors cannot provide a sufficient protection. Therefore, the two design options of OLS codes and 5RPR are not directly comparable in terms of error protection.

Table 5 shows the results for the 3RPR [23], 3MR, as well as ITDMR (inexact triple/double modular redundancy) [20] with $k = 2$ schemes under the same design specification of Tables 3 and 4. ITDMR has a moderate overhead, but it can only deal with small errors (errors on the $k + 1 = 3$ LSBs in this case). As expected, the decision circuitry for the 3RPR scheme has a higher complexity and worse performance than for a 3MR scheme; note that the advantages of using RP units (modules) for processing is not included in Tables 3, 4, 5. This aspect will be addressed in the next section. Even though the schemes of Table 5 are applicable to only a single erroneous data copy, it is interesting to note that the area complexity in *N*MR schemes $(N = 3, 5)$ grows at a faster rate than for *N*RPR, thus further confirming that at higher order of replication, a RPR scheme is better suited.

In addition, the inherent limitation of RPR schemes (i.e., small errors that only affect the LSBs, cannot be corrected) can be overcome in many applications (as already discussed in Section II); so it is interesting to compare the performance of different schemes in terms of dealing with multiple but small errors. The proposed scheme is compared against the 5MR scheme, the DEC OLS codes, and the ITDMR scheme; as an example, results for protecting 16-bit word data by truncating 2, 4-bit LSBs in the RP data (i.e., $n = 16, k = 2, 4$) when the probability of incorrect data is 1 percent are shown in Table 6. From Table 6, the existing 5MR scheme and the OLS codes can fully correct multiple small errors, while the ITDMR and the proposed schemes also have a probability of more than 98 percent to correct them.

TABLE 6. Results for performance of different schemes in terms of dealing with small errors.

Scheme		Probability of small error correction
$k = 2$	5MR	100%
	(32,16) OLS	100%
	ITDMR	99.99%
	5RPR3	98.89%
	5RPR4	99.96%
$k = 4$	5MR	100%
	(32,16) OLS	100%
	ITDMR	99.99%
	5RPR3	98.77%
	5RPR4	99.96%

Result for ITDMR is obtained as per equation (10) in [20], while the results for the proposed scheme (5RPR3 and 5RPR4) are obtained as per equations (14) and (15).

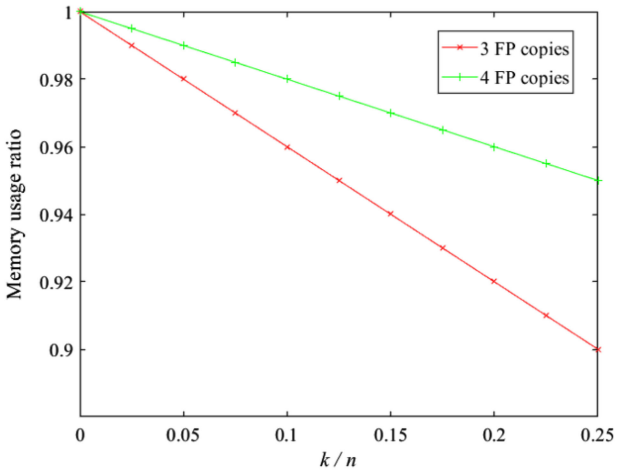

FIGURE 8. Memory usage ratio of 5RPR with different number of FP copies vs 5MR.

VI. CASE STUDIES

This section presents two case studies in which the proposed *N*RPR scheme is applied to show its effectiveness.

A. CASE 1: MEMORY

As discussed in previous sections, one or two RPR data copies can be used in the proposed 5RPR scheme to tolerate two erroneous data copies. In this first case, evaluation is pursued with respect to a memory system consisting of 5 memory chips; each memory chip provides a data copy as input to the decision circuit. Assume there are M words in a memory, and n bits per word, the total number of memory cells required by the 5RPR scheme is given as follows:

$$N_{5RPR} = (q \cdot n + (5 - q)(n - k)) \cdot M, \tag{18}$$

where $q = \{3, 4\}$ is the number of the memory chips storing FP data, and k is the number of truncated LSBs on each word.

For the 5MR scheme, the total number of required memory cells is given by:

$$N_{5MR} = 5n \cdot M. \tag{19}$$

By combining Eqs. (18) and (19), the memory usage ratio of the proposed 5RPR scheme and the 5MR scheme (N_{5RPR}/N_{5MR}) can be obtained; the results are plotted in Figure 8. The ratio decreases linearly with an increase in the number of truncated bits k. Compared to the 5MR scheme, 10 percent of the total memory can be saved when using three FP memory chips in the 5RPR scheme and by truncating 25 percent LSBs on each word in the remaining two RP chips.

B. CASE 2: DISCRETE COSINE TRANSFORM

The Discrete Cosine Transform (DCT) is widely applied in the fields of image and video coding compression by removing the correlation of image elements in the transform domain [29]. In [30], DCT uses integers in a transform matrix instead of floating point numbers. The transformation core is composed by a signed integer that achieves a high accuracy without floating-point calculation. In matrix notation, the discrete two-dimensional radix-8 DCT is given by $Y = H \cdot X \cdot H^{\mathrm{T}}$, where X is the 8x8 input image frame, H is the transform matrix and Y is the transformed (output) matrix.

DCT is implemented by two stages of matrix computation including multiplication and addition operations in signed integer format; this implementation utilizes a so-called Multiplication and Accumulation (MAC) cell design. MAC processing can be made tolerant to soft errors using the RPR scheme proposed in [25]; however, its inputs data are usually stored in memory. Hence, data must be provided correctly to the MAC hardware, else an additional error will result due to the RPR nature of the MAC system.

Therefore, in this section the proposed 5RPR scheme is used for the memory (such as already presented in the previous section), while the RPR scheme of [25] is utilized for MAC computation. The entire RPR system for DCT is shown in Figure 9. The original image information is stored in the memories employing the 5RPR scheme (as detailed in the previous section). Then the obtained D_{output} is replicated three times, and provided as input for the three copies of data D_H (as per the H matrix) to the RPR MAC scheme proposed

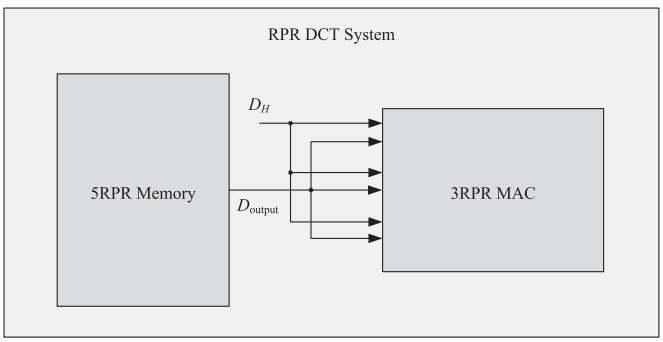

FIGURE 9. RPR DCT system.

TABLE 7. Average PSNR for images obtained in different cases.

Scheme		Best Case		Worst Case	
8-bit	$k = 2$	45.87 dB	100%	41.87 dB	91.28%
16-bit	$k = 2$	47.84 dB	100%	42.35 dB	88.52%
	$k = 4$	43.39 dB	100%	38.78 dB	89.38%

in [25]. In the MAC, two copies of D_{output} and D_H are truncated as RP input data in its implementation. The lower order of redundancy in the MAC is justified due to the lower occurrence of SEU in logic circuits compared to storage devices such as memories. Twenty 8-bit and 16-bit images from the publicly available database of [31] have been simulated using the DCT system of Figure 9. The transform matrix H of [32] (and given in equation (21)) is used in the simulations.

$$H = \begin{bmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 \end{bmatrix}. \tag{21}$$

To evaluate the effect of errors in the storage unit (memory block) on the DCT system, errors with a 100 percent error rate (i.e., at least one module is always incorrect) are analyzed; two scenarios are considered:

1) *Best case:* all errors in the memory block can be corrected (i.e., generating FP data) and thus, they have no effect on the MAC block. This case is used as baseline for the worst case;
2) *Worst case:* small errors exist on the output of the memory block (i.e., generating FP data) and thus, they affect the MAC block.

Random errors have been inserted in the memory and the MAC with a uniform distribution; 100 simulation runs have been performed for each image. The average Peak Signal Noise Ratio (PSNR) has been calculated next for all 20 images when truncating k bits in the RPR schemes.

The results for the PSNR are reported in Table 7. It can be seen that compared to the best case, the PSNR in the worst case is reduced at most by 11.48 percent for $n = 16$ and $k = 2$. Two examples of images are shown in Figure 10; differences between the best and the worst cases are hard to be distinguished by human eyes. Therefore, the proposed 5RPR scheme can be attractive in applications in which RP data can be tolerated.

VII. CONCLUSION

This paper has presented an efficient N Reduced Precision Redundancy (NRPR) scheme by using a comparison-based technique, that is used to attain error tolerance and reliable processing of data such as encountered in image processing.

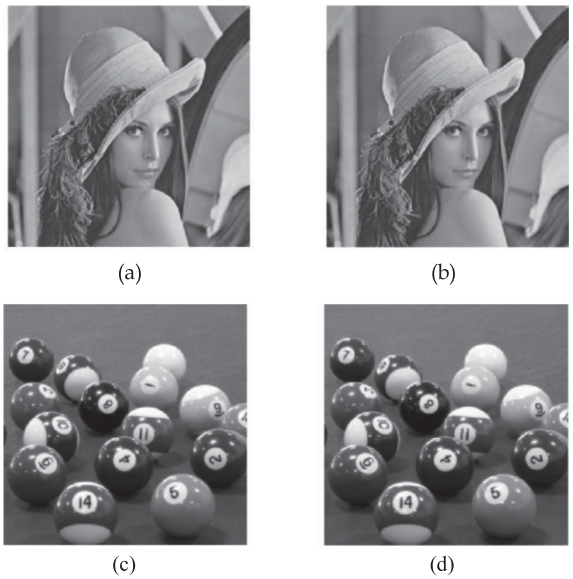

FIGURE 10. Example of images computed by the RPR DCT system: (a) best case and (b) worst case of 8-bit image, and (c) best case and (d) worst case of 16-bit image.

In particular the case of $N = 5$ copies of input data is analyzed as an example. This technique is based on a graph theoretical approach that allows the efficient design of the decision logic circuit. The decision logic circuit includes a syndrome generator and a syndrome checker. The 5RPR scheme has been shown to have practical implication as design when either 3 (2) or 4 (1) FP (RP) data copies are utilized. For these cases, a probabilistic analysis has been presented to determine the probability of having RP data as output; this analysis has shown that a low probability is accounted for the RP output even in the presence of two erroneous copies of data.

The proposed NRPR shows many advantages for the realistic case of $N = 5$; the decision circuitry has excellent performance metrics in terms of power dissipation and delay while being applicable to diverse applications such as memory and image processing. In the case of memory, redundancy management in memories with RP data results in significant savings in capacity, more pronounced when either the number of RP copies or the truncated number of bits increase. For image processing as a second application, this paper has described a RPR scheme in which tolerance is applied to both stored information (using a 5MR scheme) and its processing (using the 3RPR scheme of [25]). Results for DCT show that such a complete RPR system is very effective and output quality (measured by the PSNR) is only marginally affected by the presence of RP data and processing.

A potential problem of the proposed scheme is the presence of failures in the decision logic (e.g., a timing failure); they may cause an incorrect data copy to be provided as an output. However, for this occurrence, in addition to failure in the decision logic, at least one of the modules has to also be incorrect (as otherwise regardless of the module selected, the output would be correct). This combination of failures in

replicated modules and decision logic is not considered in this paper and left for future work.

ACKNOWLEDGMENTS

Pedro Reviriego was partially supported by the TEXEO project (TEC2016-80339-R) funded by the Spanish Research Plan. The research of Amhed Louri and Fabrizio Lombardi is supported by the USA National Science Foundation (NSF) under grant No. 1812467. Weiqiang Liu is supported partially by NSFC (61871216).

REFERENCES

[1] F. H. Gebara, H. P. Hofstee, and K. J. Nowka, "Second-generation big data systems," *IEEE Comput.*, vol. 48, no. 1, pp. 36–41, Jan. 2015.

[2] W. A. Bhat, "Is a data-capacity gap inevitable in big data storage?" *IEEE Comput.*, vol. 51, no. 9, pp. 54–62, Sep. 2018.

[3] N. Kanekawa, E. H. Ibe, T. Suga, and Y. Uematsu, *Dependability in Electronic Systems: Mitigation of Hardware Failures, Soft Errors, and Electro-Magnetic Disturbances*, New York, NY, USA, Springer Verlag, Nov. 2010.

[4] V. Vargas, P. Ramos, V. Ray, *et al.*, "Radiation experiments on a 28 nm single-chip many-core processor and SEU error-rate prediction," *IEEE Trans. Nucl. Sci.*, vol. 64, no. 1, pp. 483–490, Jan. 2017.

[5] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, "Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule," *IEEE Trans. Electron Devices*, vol. 57, no. 7, pp. 1527–1538, Jul. 2010.

[6] M. Hermann, "Radiation effects in SDRAMs," *Ionizing Radiation Effects in Electronics*, M. Bagatin and S. Gerardin, Eds. Boca Raton, FL, USA: CRC, pp. 77–95, 2016.

[7] N. T. H. Nguyen, D. Agiakatsikas, Z. Zhao, *et al.*, "Reconfiguration control networks for FPGA-based TMR systems with modular error recovery," *Microprocessors Microsyst.*, vol. 60, pp. 86–95, Apr. 2018.

[8] M. Wirthlin, D. Lee, G. Swift, and H. Quinn, "A method and case study on identifying physically adjacent multiple cell upsets using 28-nm, interleaved and SECDED-protected arrays," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 6, pp. 3080–3087, Dec. 2014.

[9] M. Maniatakos and Y. Makris, "Workload-driven selective hardening of control state elements in modern microprocessors," *Proc. 28th VLSI Test Symp.*, pp. 159–164, Apr. 2010.

[10] I. Wali, *et al.*, "A low-cost reliability vs. cost trade-off methodology to selectively harden logic circuits," *J. Electron. Testing*, vol. 33, no. 1, pp. 25–36, Feb. 2017.

[11] M. D. Gutierrez, V. Tenentes, D. Rossi, and T. J. Kazmierski, "Susceptible workload evaluation and protection using selective fault tolerance," *J. Electron. Testing*, vol. 33, no. 4, pp. 463–477, Aug. 2017.

[12] K. Namba and F. Lombardi, "Parallel decodable multi-level unequal burst error correction codes for memories of approximate systems," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3794–3801, Dec. 2016.

[13] P. Reviriego, M. Flanagan, S. Liu, and J. A. Maestro, "Multiple cell upset correction in memories using difference set codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 11, pp. 2592–2599, Nov. 2012.

[14] S. Lin and D. J. Costello, *Error Control Coding*. 2nd ed. Englewood Cliffs, NJ, USA: Prentice Hall, 2004.

[15] P. Reviriego, S. Liu, A. Sánchez-Macián, *et al.*, "A scheme to reduce the number of parity check bits in orthogonal latin square codes," *IEEE Trans. Rel.*, vol. 66, no. 2, pp. 518–528, Mar. 2017.

[16] S. Liu, P. Reviriego, A. Sánchez-Macián, J.A. Maestro, and L. Xiao, "Comments on "Extend orthogonal latin square codes for 32-bit data protection in memory applications," *Microelectron. Reliab.*, vol. 63, pp. 278–283, 2016," *Microelectronis Rel.*, vol. 69, pp. 126–129, Feb. 2017.

[17] A. Sánchez-Macián, P. Reviriego, and J. A. Maestro, "Combined SEU and SEFI protection for memories using orthogonal latin square codes," *IEEE Trans. Circuits Syst. I, Regular Papers*, vol. 63, no. 11, pp. 1933–1943, Nov. 2016.

[18] D. P. Siewiorek and R. S. Swarz, *The Theory and Practice of Reliable System Design*. Bedford, England: Digital Press, 1982.

[19] M. Augustin, M. Gossel, and R. Kraemer, "Reducing the area overhead of TMR-systems by protecting specific signals," in *Proc. IEEE 16th Int. On-Line Testing Symp.*, pp. 268–273, Jul. 2010.

[20] K. Chen, J. Han, and F. Lombardi, "Two approximate voting schemes for reliable computing," *IEEE Trans. Comput.*, vol. 66, no. 7, pp. 1227–1238, Jul. 2017.

[21] E. P. Kim and N. R. Shanbhag, "Soft N-modular redundancy," *IEEE Trans. Comput.*, vol. 61, no. 3, pp. 323–336, Mar. 2012.

[22] G. Karakonstantis, A. Chatterjee, and K. Roy, "Containing the nanometer "Pandora-Box": Cross-layer design techniques for variation aware low power systems," *IEEE J. Emerging Sel. Topics Circuits Syst.*, vol. 1, no. 1, pp. 19–29, Mar. 2011.

[23] S. Byonghyo, S. R. Sridhara, and N. R. Shanbhag, "Reliable low-power digital signal processing via reduced precision redundancy," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 12, no. 5, pp. 497–510, May 2004.

[24] A. Ullah, P. Reviriego, S. Pontarelli, and J. A. Maestro, "Majority voting-based reduced precision redundancy adders," *IEEE Trans. Device Mater. Rel.*, vol. 18, no. 1, pp. 122–124, Mar. 2018.

[25] K. Chen, L. B. Chen, P. Reviriego, and F. Lombardi, "Efficient implementations of reduced precision redundancy (RPR) multiply and accumulate (MAC)," *IEEE Trans. Comput.*, vol. 68, no. 5, pp. 784–790, Dec. 2018.

[26] H.Y. Lo, L.P. Ju, and C.C. Su, "General version of reconfiguration N modular redundancy system," *IEEE Proc. G (Circuits Devices Syst.)*, vol. 137, no. 1, pp. 1–4, Feb. 1990.

[27] F. Lombardi, "Diagnosable systems for fault tolerant computing," in *Proc. IEEE Fault Tolerant Comput. Symp.*, vol. 15, pp. 42–47, Jun. 1985.

[28] F. Lombardi, "Comparison-based diagnosis with faulty comparators," *Electron. Lett.*, vol. 22, no. 22, pp. 1158–1159, Oct. 1986.

[29] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. 23, no. 1, pp. 90–93, Jan. 1974.

[30] J. Wu and Y. Li, "A new type of integer DCT transform radix and its rapid algorithm," in *Proc. Int. Conf. Electric Inf. Control Eng.*, pp. 1063–1066, Apr. 2011.

[31] TESTIMAGES - Testing images for scientific purposes, 2017. [Online]. Available: https://sourceforge.net/projects/testimages/

[32] P. K. Meher, S. Y. Park, B. K. Mohanty, K. S. Lim, and C. Yeo, "Efficient integer DCT architectures for HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 1, pp. 168–178, Jan. 2014.