DOI: 10.1109/tvlsi.2021.3066804

Design of FPGA Implemented Reed Solomon (RS) Erasure Decoders with Fault Detection and Localization on User Memory

Zhen Gao, Lingling Zhang, Yinghao Cheng, Kangkang Guo, Anees Ullah and Pedro Reviriego

***Abstract*—Reed-Solomon erasure codes (RS-EC) are widely used in packet communication and storage systems to recover erasures. When the RS-EC decoder is implemented on a Field Programmable Gate Array (FPGA) in a space platform, it will suffer Single Event Upsets (SEUs) that can cause failures. In this paper, the reliability of an RS-EC decoder implemented on an FPGA when there are errors in the user memory is firstly studied. Then a fault detection and localization scheme is proposed based on partial re-encoding for the faults in user memory of the RS decoder. Furthermore, check bits are added in the generator matrix to improve the fault location performance. Theoretical analysis shows that the scheme could detect most faults with small missing and false detection probability. Experimental results on a case study show that more than 90% faults on user memory could be tolerated by the decoder, and the all other faults can be detected by the fault detection scheme when the number of erasures is less than the correction capability of the code. Although false alarms exist (with probability smaller than 4%), they can be used to avoid faults accumulation. Finally, the fault location scheme could accurately locate all the faults. The theoretical estimates are very close to the experiment results, which verifies the correctness of the analysis done.**

***Index Terms*—Reed Solomon decoder, Single Event Upsets (SEUs), FPGA, reliability, fault detection and location**

I. Introduction

REED-Solomon (RS) codes are efficient error correction codes that are widely used in communications and storage systems [1],[2],[3]. For applications on which some data blocks are lost and their positions are known, a variant known as RS Erasure Codes (RS-EC), can be used for data recovery [2],[4]. For example, in IP based communication networks, a message is packed into a sequence of packets and transmitted through the network, and some of them may not arrive at the destination due to network congestion. In this case, an RS-EC code could be used to introduce redundant packets at the transmitter and recover the lost packets at the receiver based on the constraints among received packets [5],[6]. For large scale storage system, data is distributed in multiple storage units (disks or servers), and a RS-EC code is usually used to introduce redundant data segments. When some of the storage units fail, the constraints among the available units could be used to recover the original data [7],[8],[9],[10].

In recent years, space based Internet has become a hot topic due to its seamless global coverage and strong robustness to failures of terrestrial infrastructures [11],[12],[13],[14]. In space based Internet, data would be transmitted as IP packets and large amounts of data will be stored on the satellite platform [13],[14], so RS-EC could be widely used for reliable data transmission and storage. But differently from the application on terrestrial systems, cosmic radiation may cause the failure of the on-board digital electronic systems [15],[16], so the reliability of the RS encoder and decoder becomes an important problem, especially for the decoder whose complexity is much higher than the encoder. On the other hand, SRAM based FPGAs (SRAM-FPGAs) are a popular option for on-board digital processing due to their rich logic resources and good re-configurability [17],[18]. However, SRAM-FPGAs are more sensitive to cosmic radiation, and Single Event Upsets (SEUs) are the most frequent events [18],[19]. SRAM-FPGAs can suffer two types of SEUs: errors on the configuration memory and errors on the user memory. The errors on the configuration memory may change the design and they are permanent unless the FPGA is reconfigured [19]. For user memory, e.g. registers and BRAMs, the SEU may modify the parameters and intermediate variables and corrupt the results. Therefore, if RS codes are implemented in an SRAM-FPGA on a space platform, the reliability of the decoder would be an important issue [20], and the protection of RS decoders against errors induced by SEUs becomes important.

There are several works on the protection of the decoder for RS error correcting codes against SEUs. In [21], the traditional triple redundancy modular (TMR) method is applied to protect the flip-flops and the clock and reset signaling in the encoder and decoder. The authors in [22] proposed to detect faults in the RS decoder by checking whether the output is a valid codeword. Instead, the work presented in [23] discussed the same self-checking property of RS decoder, and proposed to reduce the complexity of the fault detection logic with a small loss of the fault tolerance capability. However, as far as the authors know,

Manuscript received on Nov.10, 2020. This work was supported in part by the National Natural Science Foundation of China (61501321), in part by the China Postdoctoral Science Foundation and Luoyang Newvid Technology Co., Ltd,, and in part by the ACHILLES project PID2019-104207RB-I00 funded by the Spanish Ministry of Science and Innovation.

Z. Gao, L. Zhang, Y. Cheng and K. Guo are with the Tianjin University, Tianjin 300072, China (e-mail: {zgao, zhanglllling, yhc, guokk}@tju.edu.cn).

A. Ullah is with University of Engineering and Technology, Peshawar, Abbottabad Campus, Abbottabad 220101, Pakistan (e-mail: aneesullah@uetpeshawar.edu.pk

P. Reviriego is with Universidad Carlos III de Madrid, 28911 Leganés, Spain (e-mail: revirieg@it.uc3m.es).

there is no research aiming at the SEU detection and localization for the decoder of RS-EC.

In this paper, an efficient fault detection and localization scheme is proposed to detect the faults induced by SEUs and to locate their position in the user memory of the RS-EC decoder. The SEU detection and location information could be used by the fault management logic of the system for efficient recovery.

The rest of the paper is organized as follows. In Section II, the principle of RS-EC encoder and decoder is introduced. In Section III, the reliability of the user memory in the RS-EC decoder to SEUs is analyzed theoretically and is verified by hardware based fault injection experiments. In Section IV, a scheme for fault detection and localization in the RS-EC decoder is proposed, and the evaluation is conducted in Section V. Finally, the paper is concluded in Section VI.

II. Overview of an RS Erasure Decoder

In this section, a brief overview of the RS-EC encoder and decoder is provided, and then the matrix inversion in RS-EC decoder is briefly described.

A. Encoder and Decoder for RS-EC

An (k, m) RS-EC is composed of k data symbols ($\boldsymbol{d} = [d_1, d_2, \ldots, d_k]^T$) and m parity symbols ($\boldsymbol{p} = [p_1, p_2, \ldots, p_m]^T$). Each symbol is expressed as a w-bit word, and the combination of the $k+m$ symbols is defined as a codeword $\boldsymbol{c} = [\boldsymbol{d}^T \boldsymbol{p}^T]^T$ on the Galois Field GF(2^w). Any combination of k symbols (data or parity) could be used to recover the original k data symbols [5]. In other words, the maximum number of erasures is m.

In the encoder, the m parity symbols of RS-EC can be generated from k data symbols by matrix multiplication over GF(2^w) [4] as follows:

$$\boldsymbol{p} = \boldsymbol{G} \times \boldsymbol{d} \tag{1}$$

where $\boldsymbol{G}$ is an $m \times k$ generator matrix defined over GF(2^w), and is commonly constructed in the form of a Vandermonde matrix [24] as shown in the next equation.

$$\boldsymbol{G} = \begin{bmatrix} 1^0 & 2^0 & \cdots & k^0 \\ 1^1 & 2^1 & \cdots & k^1 \\ \vdots & \vdots & \ddots & \vdots \\ 1^{m-1} & 2^{m-1} & \cdots & k^{m-1} \end{bmatrix} \tag{2}$$

The decoder of an (k, m) RS-EC could recover data symbols as follows. For the case that e data symbols are erased ($e \le m$), we can construct an $e\times 1$ vector $\boldsymbol{p_e}$ of e available parity symbols, a $k\times 1$ vector $\boldsymbol{d_e}$ of the available data symbols and 0s on the positions of erasures, an $e\times k$ matrix $\boldsymbol{G'}$ with e rows from $\boldsymbol{G}$ corresponding to the e available parity symbols in $\boldsymbol{p_e}$, and an $e\times e$ matrix $\boldsymbol{G''}$ with e columns of $\boldsymbol{G'}$ corresponding to the e erased data symbols. Then the vector of erased data symbols $\boldsymbol{d_r}$ could be recovered according to equations (3) and (4) as explained in [6], and the process is shown in Fig. 1 for an (3, 3) RS-EC.

$$\boldsymbol{p_I} = \boldsymbol{p_e} - \boldsymbol{G'} \times \boldsymbol{d_e} \tag{3}$$

$$\boldsymbol{d_r} = (\boldsymbol{G''})^{-1} \times \boldsymbol{p_I} \tag{4}$$

Since the code is defined over Galois Field GF(2^w), the operation of addition (or subtraction) between elements is actually performed as XOR, and the multiplication (or division) operation is performed based on a logarithmic table (*gflog*) and an inverse logarithmic table (*gfilog*) on GF(2^w) as explained in [8] and shown in the next equation.

$$x \times y = gfilog\left(gflog(x) + gflog(y)\right) \tag{5}$$

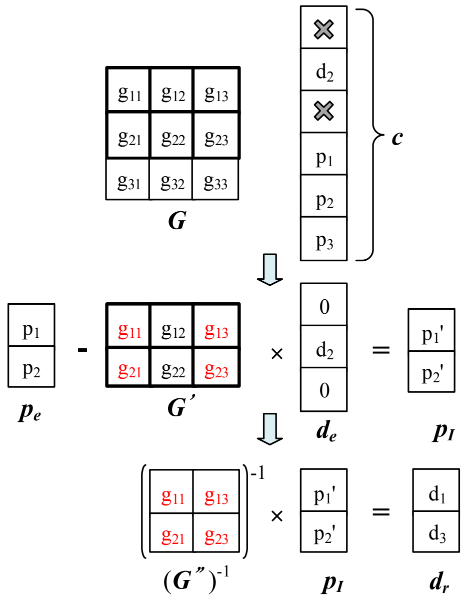

Fig.1. RS-EC decoding procedure (example for an (3,3) RS-EC)

B. Matrix Inversion based on LU Decompostion

The main complexity of the RS-EC decoder comes from the matrix inversion of $\boldsymbol{G''}$ in equation (4). In practice, LU decomposition is a commonly used method for matrix inversion [26]. As shown in equation (6), a nonsingular $n\times n$ square matrix $\boldsymbol{A}$ could be decomposed by LU decomposition into the form of $\boldsymbol{A}=\boldsymbol{L}\times\boldsymbol{U}$, where $\boldsymbol{L}$ is a unit lower triangular matrix and $\boldsymbol{U}$ is an upper triangular matrix [27].

$$\underbrace{\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} 1 & & \\ \vdots & \ddots & \\ l_{n1} & \cdots & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} u_{11} & \cdots & u_{1n} \\ & \ddots & \vdots \\ & & u_{nn} \end{pmatrix}}_{U} \tag{6}$$

The elements in matrix $\boldsymbol{L}$ and $\boldsymbol{U}$ could be calculated as [27],[28]:

$$\begin{cases} u_{1j} = a_{1j} & (j = 1,2,\cdots,n); \\ l_{i1} = a_{i1}/u_{11} & (i = 2,3,\cdots,n); \\ u_{kj} = a_{kj} - \sum_{t=1}^{k-1} l_{kt}u_{tj} & (j = k,k+1,\cdots,n; k = 2,3,\cdots,n); \\ l_{ik} = \dfrac{1}{u_{kk}}\left(a_{ik} - \sum_{t=1}^{k-1} l_{it}u_{tk}\right) & (i = k+1,k+2,\cdots,n; k = 2,3,\cdots,n). \end{cases} \tag{7}$$

Then the elements of $\boldsymbol{U}^{-1}$ can be obtained as [29]:

$$\begin{cases} v_{ii} = 1/u_{ii} & (i = 1,2,\cdots,n); \\ v_{ij} = -\dfrac{1}{u_{ii}} \sum_{k=i+1}^{j} u_{ik}v_{kj} & (i = n-1,n-2,\cdots 1; j = i+1,\cdots,n). \end{cases} \tag{8}$$

Following the same procedure, we can get the inverse of $\boldsymbol{L}^T$ (the transposed matrix of $\boldsymbol{L}$), and then obtain $\boldsymbol{L}^{-1}$ according to:

$$\boldsymbol{L}^{-1} = ((\boldsymbol{L}^T)^{-1})^T \tag{9}$$

Finally, the inverse matrix of $\boldsymbol{A}$ can be calculated as:

$$\boldsymbol{A}^{-1} = \boldsymbol{U}^{-1} \times \boldsymbol{L}^{-1} \tag{10}$$

III. Reliability of the Unprotected RS-EC Decoder to SEUs on User Memory

In this section, a general FPGA implementation of an RS-EC decoder is described and the reliability on each part of the user memory of the decoder is analyzed theoretically and evaluated by fault injection experiments.

A. A general FPGA Implementation of RS-EC Decoder

A simplified decoder for the (k,m) RS-EC was implemented using Verilog and mapped on a Xilinx Zynq 7000 SoC (xc7z 030ffg676-1). The LU decomposition-based matrix inversion is applied for equation (4). As shown in Table I, the user memories of the RS-EC decoder include all the matrices and vectors for equations (2), (3) ~ (4) and (6) ~ (9), and are stored in Block RAMs and registers. An additional vector $\boldsymbol{p_{ed}}$ is used to hold the positions for the erased data symbols. The value of k, m and w could be set in the top module, and the space for all matrices and vectors is pre-allocated for the worst case of m erasures. From the table, the total user memory for the (k,m) RS-EC decoder includes $N = k+4m+2mk+2(2^w-1)+8m^2$ words over GF(2^w), which corresponds to $N_b = wN$ bits. Among all the memories, the generator matrix $\boldsymbol{G}$, logarithmic table (*gflog*) and inverse logarithmic table (*gfilog*) are static and correspond to $w(mk+2(2^w-1))$ bits, and others corresponding to $w(k+4m+mk+8m^2)$ bits would be recalculated and refreshed for each decoding.

TABLE I. User Memory of the RS Decoder

User Memory (x)		Size (S_x)	Notes
$\boldsymbol{G}$		$m\times k$	Generator matrix
Gflog		2^w-1	Logarithm table
Gfilog		2^w-1	Inverse logarithm table
$\boldsymbol{p_{ed}}$		$m\times 1$	Positions of erased data symbols
$\boldsymbol{d_e}$		$k\times 1$	Data symbols with 0s for erased
$\boldsymbol{G'}$		$m\times k$	Subset of G
$\boldsymbol{p_e}$		$m\times 1$	Available parity symbols
$\boldsymbol{p_I}$		$m\times 1$	Intermediate vector
$\boldsymbol{G''}$		$m\times m$	Subset of G'
$(\boldsymbol{G''})^{-1}$		$m\times m$	Inverse matrix of G''
$\boldsymbol{d_r}$		$m\times 1$	Recovered data symbols
Matrix inversion based on LU decomposition	$\boldsymbol{L}$	$m\times m$	Unit lower triangular matrix
	$\boldsymbol{U}$	$m\times m$	Upper triangular matrix
	$\boldsymbol{L}^{\mathrm{T}}$	$m\times m$	Transposed matrix of L
	$(\boldsymbol{L}^{\mathrm{T}})^{-1}$	$m\times m$	Inverse matrix of L^{T}
	$\boldsymbol{L}^{-1}$	$m\times m$	Inverse matrix of L
	$\boldsymbol{U}^{-1}$	$m\times m$	Inverse matrix of U

B. Reliability Analysis of the User Memory of RS-EC decoder

In this paper, the reliability of RS-EC decoder against SEUs on the user memory is evaluated by the SEU tolerance rate, which is the probability that a SEU on a bit would not change the decoding results for different inputs. The evaluation is performed for the case that the erased data symbols could be recovered correctly, so following two cases are not considered: 1) no data symbols are erased (decoding is not needed); 2) the number of erased symbols is larger than m. Defining the number of erased symbols as E ($1\leq E\leq m$), including e data symbols and e_p parity symbols ($E = e + e_p$), the reliability of each part of user memory to SEUs is analyzed in the following.

(1) Analysis for SEUs on the generator matrix $\boldsymbol{G}$

For e erased data symbols, e rows in the generator matrix $\boldsymbol{G}$ will be selected to construct $\boldsymbol{G'}$. Therefore, if a word in one of these e rows is corrupted by SEU, the decoding results would be wrong, and SEUs on other rows would not affect the results. Since the probability for e data symbols among E erasures is:

$$P(E,e)=\frac{C_k^e C_m^{e_p}}{C_{k+m}^E - C_m^E} \tag{11}$$

the SEU tolerance rate for the case of e erased data symbols could be expressed as:

$$P_t(E,e)=\frac{m-e}{m}P(E,e) \tag{12}$$

Then the average tolerance rate for SEUs on $\boldsymbol{G}$ for E erased symbols could be calculated as:

$$P_t^G(E)=\sum_{e=1}^{E}P_t(E,e) \tag{13}$$

The effect of SEUs on variables $\boldsymbol{G'}$, $\boldsymbol{p_e}$, $\boldsymbol{p_I}$, $\boldsymbol{d_r}$ and $\boldsymbol{p_{ed}}$ could be analyzed in the same way, and the corresponding average SEU tolerance rate can also be calculated based on formula (13). It should be noted that the estimated SEU tolerance rate is for the effective period of these matrix or vector, that is between the moment the matrix or vector is refreshed and the moment they have been used for decoding of the current input. Any SEU out of the effective period will not cause decoding errors.

(2) Analysis for SEUs on logarithm table *gflog*

The reliability of the logarithm table *gflog* and the inverse logarithm table *gfilog* depends on the number of different words that are used during the RS decoding process. Table *gflog* is used for the operands of the multiplication and division operation in equations (3) and (4) and the matrix inversion $(\boldsymbol{G''})^{-1}$ based on equations (7) and (8). The number of different operands for each of these equations is analyzed as follows.

In equation (3), *gflog* is used for $\boldsymbol{G'}\times\boldsymbol{d_e}$. For the case that e data symbols are erased, the number of non-zero symbols in $\boldsymbol{d_e}$ is $N_d = k-e$. As for the $m\times k$ matrix $\boldsymbol{G'}$, $e\times k$ non-zero elements are used during the matrix multiplication, but only part of them need to be looked up in *gflog* because the first row and first column of $\boldsymbol{G}$ are all 1s. In this case, the number of elements in $\boldsymbol{G'}$ that need to be looked up in *gflog* is related to the rows of $\boldsymbol{G}$ that are selected to form $\boldsymbol{G'}$, and is mainly determined by whether the first data symbol (d_1) and the first parity symbol (p_1) is erased. There are four cases regarding whether d_1 or p_1 is erased. In Table II, the 3rd column lists the probability for each case (p), and the corresponding number of elements in $\boldsymbol{G'}$ that need to be looked up from *gflog* (n) are listed in 4th column. Finally, the expected number of elements in *gflog* that would be used for $\boldsymbol{G'}$ could be calculated as

$$N_{G'}(E,e)=\sum_{i=1}^{4}p_i(E,e)n_i \tag{14}$$

TABLE II. Some Numbers for Analysis of gflog and gfilog

Idx i	d_1/p_1		Probability $p_i(E, e)$	Elements in (3) n	Repeats (*gflog*) r^{log}	Repeats (*gfilog*) r^{ilog}
1	√	√	$\frac{C_{k-1}^{e}C_{m-1}^{e_p}}{C_k^e C_m^{e_p}}$	(e-1)*(k-e-1)+1	3e-2	(5e^2-3e+2)/2
2	×	√	$\frac{C_{k-1}^{e-1}C_{m-1}^{e_p}}{C_k^e C_m^{e_p}}$	(e-1)*(k-e)+1	4e-3	3e^2-2e+1
3	√	×	$\frac{C_{k-1}^{e}C_{m-1}^{e_p-1}}{C_k^e C_m^{e_p}}$	e*(k-e-1)+1	e-1	0
4	×	×	$\frac{C_{k-1}^{e-1}C_{m-1}^{e_p-1}}{C_k^e C_m^{e_p}}$	e*(k-e)	3e-2	(5e^2-5e+2)/2

For equation (4), e elements in $\boldsymbol{p_I}$ ($N_p = e$) and e^2 elements in $(\boldsymbol{G''})^{-1}$ ($N_{invG''} = e^2$) need to be looked up from *gflog*. For the matrix inversion $(\boldsymbol{G''})^{-1}$, *gflog* is used for each element in matrix $\boldsymbol{G''}$, $\boldsymbol{L}$, $\boldsymbol{U}$, $\boldsymbol{L}^{-1}$, and $\boldsymbol{U}^{-1}$. The number of *gflog* look-ups for each of these matrices is analyzed as follows.

(a) The first row and first column of $\boldsymbol{G''}$ are used for multiplication and division operations, so $N_{G''} = 2e\text{-}1$ *gflog* look-ups are required.

(b) Matrix $\boldsymbol{L}$ has $e(e+1)/2$ non-zero elements. Except the 1s on the diagonal, the number of elements that need to be looked up from *gflog* is $N_L = e(e+1)/2\text{-}e = e(e\text{-}1)/2$.

(c) Matrix $\boldsymbol{U}$ has $e(e+1)/2$ non-zero elements. Since the first row is same to that of $\boldsymbol{G''}$, the number of elements that need to be looked up from *gflog* is $N_U = e(e+1)/2\text{-}e = e(e\text{-}1)/2$.

(d) Matrix $\boldsymbol{L}^{-1}$ has $e(e+1)/2$ non-zero elements, and the ones on the diagonal are 1s, so the number of elements that need to be looked up from *gflog* is $N_{invL} = e(e+1)/2\text{-e} = e(e\text{-}1)/2$.

(e) Matrix $\boldsymbol{U}^{-1}$ has $e(e+1)/2$ non-zero elements and is used to multiply $\boldsymbol{L}^{-1}$ for $(\boldsymbol{G''})^{-1}$. Since the e-th column of $(\boldsymbol{G''})^{-1}$ is the same as that of $\boldsymbol{U}^{-1}$ because the diagonal elements of $\boldsymbol{L}^{-1}$ are 1s, the number of *gflog* look-ups for this part should be $N_{invU} = e(e+1)/2\text{-}e = e(e\text{-}1)/2$.

As a result, the total number of *gflog* look-ups during the calculation of $(\boldsymbol{G''})^{-1}$ is $N_{inv} = N_{G''} + N_L + N_U + N_{invL} + N_{invU} = 2e^2\text{-}1$. But some elements are counted several times, and the number is related to whether the first data symbol (d_1) and the first parity symbol (p_1) is erased. The numbers of repeats for the four possible combinations are listed in the 5th column of Table II ($r^{\log}$), and the final number of *gflog* look-ups for the calculation of $(\boldsymbol{G''})^{-1}$ could be amended as:

$$N'_{inv}\left(E,e\right)=\sum\nolimits_{i=1}^{4}p_i\left(E,e\right)\left(N_{inv}-r_i^{\log}\right) \tag{15}$$

In summary, the total number of different *gflog* look-ups during the decoding process with E erased symbols and e data erasures could be estimated as:

$$N_{\log}\left(E,e\right)=N_d+N_{G'}\left(E,e\right)+N_p+N_{invG''}+N'_{inv}\left(E,e\right), \tag{16}$$

which is also the number of elements that SEUs on them would cause decoding errors. Since there are 2^w-1 elements in *gflog*, the average SEU tolerance rate of *gflog* for E erased symbols can be estimated as:

$$P_t^{\log}(E)=\sum_{e=1}^{E}\frac{2^w-1-N_{\log}(E,e)}{2^w-1}P(E,e)\cdot \tag{17}$$

(3) Analysis for SEUs on inverse logarithm table *gfilog*

Table *gfilog* is used on the results for multiplication and division operations in equations (3) and (4) and those used during the matrix inversion $(\boldsymbol{G''})^{-1}$. The number of *gfilog* look-ups for each procedure is analyzed as follows.

In equation (3), the vector $\boldsymbol{d_e}$ includes k-e non-zero elements for e data erasures, so $e(k\text{-e})$ multiplications are involved in $\boldsymbol{G'}\times\boldsymbol{d_e}$, which is also the number of *gfilog* look-ups ($N_3 = e(k\text{-e})$). Similarly, the produce of $(\boldsymbol{G''})^{-1}$ (with e^2 non-zero elements) and $\boldsymbol{p_I}$ (with e non-zero elements) requires e^2 table look-ups from *gfilog* for equation (4) ($N_4 = e^2$).

To get the matrix inversion $(\boldsymbol{G''})^{-1}$, *gfilog* table is used for each element in matrix $\boldsymbol{L}$, $\boldsymbol{U}$, $\boldsymbol{L}^{-1}$, $\boldsymbol{U}^{-1}$ and $(\boldsymbol{G''})^{-1}$. According to formulas (7) and (8), the number of *gfilog* look-ups for each matrix is calculated as follows:

(a) According to formula (7), the generation of the first column of matrix $\boldsymbol{L}$ requires e-1 divisions. For another column c ($2 \le c \le e\text{-}1$), c-1 multiplications and 1 division are required. So the number of *gfilog* look-ups for matrix $\boldsymbol{L}$ can be expressed as:

$$N_{7L}=e-1+\sum_{c=2}^{e-1}c(e-c) \tag{18}$$

(b) According to formula (7), the elements of first row of matrix $\boldsymbol{U}$ do not need to be calculated. For other columns, r-1 multiplications are required for e+1-r elements in the r-th row ($2 \le r \le e$). So the number of *gfilog* look-ups for matrix $\boldsymbol{U}$ can be expressed as:

$$N_{7U}=\sum_{r=2}^{e}(r-1)(e+1-r) \tag{19}$$

(c) According to formula (8), the diagonal elements of $\boldsymbol{U}^{-1}$ need e divisions. For other elements, the one in the r-th row and the c-th column ($1 \le r \le e\text{-}1$, $r+1 \le c \le e$) requires c-r multiplications and 1 division. So, the number of *gfilog* look-ups for matrix $\boldsymbol{U}^{-1}$ can be expressed as:

$$N_{8invU}=e+\sum_{r=1}^{e-1}\sum_{c=r+1}^{e}(c-r+1) \tag{20}$$

(d) The calculation of $\boldsymbol{L}^{-1}$ is similar as that of $\boldsymbol{U}^{-1}$ except that the diagonal elements are 1s, So the number of *gfilog* look-ups for matrix $\boldsymbol{L}^{-1}$ is $N_{8invL} = N_{8invU} - e$.

(e) Both $\boldsymbol{U}^{-1}$ and $\boldsymbol{L}^{-1}$ are triangular matrices, so the number of non-zero multiplications during $\boldsymbol{U}^{-1}\times\boldsymbol{L}^{-1}$ is:

$$N_{iUiL}=\sum_{r=1}^{e}(1+2+\cdots+r+(e-r)r)=\sum_{r=1}^{e}(e+\frac{1-r}{2})r \tag{21}$$

As a result, the total *gfilog* look-ups during matrix inversion $(\boldsymbol{G''})^{-1}$ is $M_{inv} = N_{7L} + N_{7U} + N_{8invL} + N_{8invU} + N_{iUiL}$. Like that for *gflog* look-ups, the number of different *gfilog* look-ups needs to be amended according to whether the first data symbol and the first parity symbol is erased. The decrements for the four cases are listed in the last column of Table II (r^{ilog}), and the amended number of *gfilog* look-ups for $(\boldsymbol{G''})^{-1}$ is:

$$M'_{inv}\left(E,e\right)=\sum\nolimits_{i=1}^{4}p_i\left(E,e\right)\left(M_{inv}-r_i^{\mathrm{ilog}}\right) \tag{22}$$

In summary, the total number of different *gfilog* look-ups during the decoding process with E erased symbols and e data erasures could be estimated as:

$$N_{\mathrm{ilog}}\left(E,e\right)=N_3+N_4+M'_{inv}\left(E,e\right) \tag{23}$$

Since there are 2^w-1 elements in *gfilog*, the average SEU tolerance rate of *gfilog* for E erasures can be estimated as:

$$P_t^{\mathrm{ilog}}(E)=\sum_{e=1}^{E}\frac{2^w-1-N_{\mathrm{ilog}}(E,e)}{2^w-1}P(E,e) \tag{24}$$

(4) Analysis for SEUs on vector $\boldsymbol{d_e}$

The vector $\boldsymbol{d_e}$ is the basis for recovery of the erased data symbols. So SEU on any element in $\boldsymbol{d_e}$ will change the decoding results, and the SEU tolerance rate for $\boldsymbol{d_e}$ is $P_t^{de} = 0$.

(5) Analysis for SEUs on matrix $\boldsymbol{G''}$ and $(\boldsymbol{G''})^{-1}$

For e data erasures, both $\boldsymbol{G''}$ and $(\boldsymbol{G''})^{-1}$ includes e^2 non-zero elements, and SEUs on other elements would not cause decoding errors. So, the average SEU tolerance rate for $\boldsymbol{G''}$ and $(\boldsymbol{G''})^{-1}$ during their effective period could be estimated as:

$$P_t^{G''}\left(E\right)=P_t^{invG''}\left(E\right)=\sum_{e=1}^{E}\frac{m^2-e^2}{m^2}P(E,e) \tag{25}$$

(6) Analysis for SEUs on intermediate matrices for $(\boldsymbol{G''})^{-1}$

There are 6 blocks of memories involved during the LU-decomposition based matrix inversion $(\boldsymbol{G}'')^{-1}$, corresponding to matrix $\boldsymbol{L}$, $\boldsymbol{U}$, $\boldsymbol{L}^{\mathrm{T}}$, $\boldsymbol{L}^{-1}$, $\boldsymbol{U}^{-1}$ and $(\boldsymbol{L}^{\mathrm{T}})^{-1}$, respectively.

For e data erasures, e^2 non-zero elements in $(\boldsymbol{G}'')^{-1}$ need to be calculated, so the number of useful elements in $\boldsymbol{L}$, $\boldsymbol{U}$ and $\boldsymbol{L}^{\mathrm{T}}$ are all $e{\bullet}(e+1)/2$. SEUs on other elements would not cause decoding errors because they can be corrected during the triangular matrix inversion. Therefore, for a total of E erasures, the average SEU tolerance rate for these three matrices during their effective period can be expressed as:

$$P_t^L = P_t^U = P_t^{L^T} = \sum_{e=1}^{E} \frac{m^2 - e(e+1)/2}{m^2} P(E,e) \tag{26}$$

For triangular inverse matrices $\boldsymbol{L}^{-1}$, $\boldsymbol{U}^{-1}$ and $(\boldsymbol{L}^{\mathrm{T}})^{-1}$, they all have e^2 useful elements, so SEUs on other m^2-e^2 memory words would not cause decoding errors. Therefore, for a total of E erasures, the average SEU tolerance rate for these inversion matrices during their effective period can be expressed as:

$$P_t^{invL} = P_t^{invU} = P_t^{invL^T} = \sum_{e=1}^{E} \frac{m^2 - e^2}{m^2} P(E,e) \tag{27}$$

(7) Total average SEU tolerance rate

Based on above analysis, the total average tolerance rate for SEUs on all the memories of the RS-EC decoder during the whole decoding period could be estimated as

$$P_t^T(E) = \frac{1}{N} \sum_x S_x \left(1 - q_x \left(1 - P_t^x\right)\right) \tag{28}$$

in which x denotes one part of the user memories listed in Table I, and S_x, P_t^x and q_x denote the number of words, SEUs tolerance rate during the effective period and the portion of the effective period over the whole decoding time, respectively.

C. Fault Injection Experiment Results

To verify the theoretical analysis, fault injection experiments were conducted on the decoder for an (6, 3) RS-EC over GF(2^8) (k = 6, m = 3, w = 8). In this case, the size of *gflog* and *gfilog* are both 255, and the total number of memory words (N) is 636 according to the analysis in Section III.A. The fault injection platform and the test procedure will be introduced in detail in Section V.B. For each part of user memory, SEU injection is conducted separately for the case of 1, 2 or 3 erasures (E = 1, 2 or 3). For each case, random inputs are generated for the encoder, and the erasure positions are randomly selected. For the SEU on a memory bit in each cycle, 100 runs of the decoding are performed for 100 different inputs to average the SEU fault tolerance rate. So for the memory x with S_x words, the total number of SEU injections is $N_I^x = 3{\times}100{\times}S_x{\times}8{\times}T_d$, where T_d is the total decoding time in cycles and is different for different erasure cases. Then the SEU tolerance rate for memory x with E erasures is measured as $O_t^x(E) = N_t^x(E)/N_I^x$, where $N_t^x(E)$ is the number of injections on memory x that do not change the decoding results. The experiment results of SEU tolerance rate for different number of erasures are listed in Table III, in which the total SEU tolerance rate in the last row is the weighted average over all memories as $\frac{1}{N}\sum_x S_x O_t^x(E)$. The theoretical estimations are also provided for comparison based on equation (28), in which the effective portion for memory x (q_x) is measured based on the implementation and listed in Table IV.

TABLE III. THEORETICAL AND EXPERIMENTAL SEU TOLERANCE RATES FOR USER MEMORY IN RS DECODER

E / Mem x	1		2		3	
	Theory	Expe	Theory	Expe	Theory	Expe
G	93.96%	93.96%	91.36%	91.40%	88.48%	88.45%
gflog	97.08%	97.44%	95.25%	96.04%	92.70%	93.40%
gfilog	97.82%	97.92%	96.24%	96.69%	92.84%	93.83%
$\boldsymbol{p_{ed}}$	97.11%	97.11%	95.09%	95.08%	91.90%	91.81%
$\boldsymbol{d_e}$	8.66%	9.04%	8.22%	8.57%	7.54%	7.66%
G'	98.95%	98.95%	96.39%	96.39%	91.90%	91.92%
$\boldsymbol{p_e}$	93.96%	93.96%	89.35%	89.18%	82.86%	82.84%
$\boldsymbol{p_I}$	78.74%	78.74%	68.50%	68.54%	54.70%	53.73%
G''	92.83%	92.83%	82.74%	82.75%	65.68%	65.36%
$(\boldsymbol{G}'')^{-1}$	98.95%	98.95%	97.64%	97.70%	95.84%	95.83%
$\boldsymbol{d_r}$	97.38%	97.38%	96.38%	96.44%	95.37%	95.30%
L	98.51%	98.51%	97.10%	97.09%	94.82%	94.79%
U	94.31%	94.31%	89.23%	89.22%	80.99%	80.66%
$\boldsymbol{L}^{\mathrm{T}}$	94.23%	94.23%	89.00%	88.97%	80.65%	80.49%
$(\boldsymbol{L}^{\mathrm{T}})^{-1}$	97.03%	97.03%	92.91%	92.97%	86.00%	85.97%
$\boldsymbol{L}^{-1}$	96.15%	96.15%	90.95%	91.15%	82.52%	82.42%
$\boldsymbol{U}^{-1}$	95.98%	95.98%	90.55%	90.38%	81.97%	82.25%
Total	96.29%	**96.47%**	94.15%	**94.65%**	90.56%	**91.22%**

TABLE IV. EFFECTIVE PORTION FOR USER MEMORY OF RS(6,3) DECODER

E / Mem x	1	2	3
G	18.11%	17.82%	17.08%
gflog	92.91%	93.27%	93.83%
gfilog	92.91%	93.27%	93.83%
$\boldsymbol{p_{ed}}$	8.66%	10.13%	12.00%
$\boldsymbol{d_e}$	91.34%	91.78%	92.46%
G'	3.15%	7.44%	12.00%
$\boldsymbol{p_e}$	18.11%	21.96%	25.41%
$\boldsymbol{p_I}$	63.78%	64.98%	67.14%
G''	64.57%	65.73%	67.83%
$(\boldsymbol{G}'')^{-1}$	9.45%	8.97%	8.23%
$\boldsymbol{d_r}$	7.87%	7.47%	6.86%
L	13.39%	13.66%	14.16%
U	51.18%	50.80%	52.01%
$\boldsymbol{L}^{\mathrm{T}}$	51.97%	51.87%	52.94%
$(\boldsymbol{L}^{\mathrm{T}})^{-1}$	26.77%	27.00%	27.67%
$\boldsymbol{L}^{-1}$	34.65%	34.48%	34.54%
$\boldsymbol{U}^{-1}$	36.22%	35.97%	35.63%

From Table III we can see that the theoretical estimations match the experimental results very well, and two important conclusions are revealed. First, the total SEU tolerance is higher than 90%, which means most of the SEUs on user memories could be tolerated by the RS-EC decoder itself. The main contribution to the overall high reliability comes from the logarithm table and the inverse logarithm table. These two tables account 80% of all the memories and are effective during over 90% of the decoding time. However, only very small portion of the elements in the tables are used for each decoding. Second, the reliability of the RS-EC decoder decreases when the number of erasures increases. This is reasonable because when the number of erasures increases, more elements in the two tables and the pre-assigned memory for the matrices and vectors would be effective, so the SEUs have higher probability to change the decoding results.

IV. Detection and Localization of SEUs on User Memories in RS-EC Decoder

This section describes the scheme proposed to detect and locate SEUs on the user memory. Detection is done by checking the consistency of the parity symbols by partial re-encoding and subsequently location is achieved by re-decoding. Finally, parity check bits are used to protect the generator matrix.

A. Basic Fault Detection and Location Scheme

(1) Fault Detection based on Partial Re-encoding

Since the complexity of RS-EC encoder is much lower than that of the decoder, we proposed to embed a partial RS-EC encoder in the decoder to detect the SEUs on the user memories. For the case that m_r parity symbols $\boldsymbol{p_r}$ are received at the decoder ($m_r = m-e_p$), we only generate m_r new parity symbols corresponding to the received parity symbols by $\boldsymbol{p'_r} = \boldsymbol{G_p} \times \boldsymbol{d'}$, where $\boldsymbol{d'}$ is the $k \times 1$ data symbol vector including the received and recovered ones, and $\boldsymbol{G_p}$ is a $m_r \times k$ matrix formed by the m_r rows of $\boldsymbol{G}$ corresponding to the m_r received parity symbols. Then the newly generated parity symbols $\boldsymbol{p'_r}$ are compared with the received ones $\boldsymbol{p_r}$. If all the parity symbols are the same, the RS-EC decoder is considered as fault-free. Otherwise, there should be an SEU on the user memory. The partial encoding uses the same user memory for generator matrix and *gflog*/ *gfilog* tables in the decoder. Initial analysis shows that this scheme is effective for detection of most SEUs on *gflog*/*gfilog* tables and all SEUs on the intermediate variables listed in Table I, but SEUs on the generator matrix and some SEUs on *gflog*/*gfilog* tables may cause missing detection or false alarm. For the former, when the corrupted element in the generator matrix or *gflog*/*gfilog* tables causes wrong recovered data symbols, it is possible to get the same parity symbol with the same wrong data symbols and corrupted elements in *gflog* or *gfilog* tables. The latter event mainly happens when a corrupted element in the generator matrix or *gflog*/*gfilog* tables is not used during decoding but used during re-encoding. In addition, the SEUs on $\boldsymbol{p'_r}$ may also cause false alarm.

(2) Fault Location based on Re-decoding

As we introduced in Section III.A, among all the user memories, the generator matrix $\boldsymbol{G}$, logarithmic table *gflog* and inverse logarithmic table *gfilog* are static, and all other matrices and vectors would be refreshed for a new decoding. So, when a fault is reported by the detection scheme, we propose to repeat the decoding and compare the outputs with those of the first decoding. If the two outputs are different, we can confirm that SEUs happen on the intermediate memories during the first decoding. In this case, the outputs of the second decoding should be correct, and no repairing is needed. If the two outputs are the same, the fault should come from the static part of the memories. In this case, it cannot be further identified whether the fault is on the generator matrix or on *gflog*/*gfilog* tables, so the fault needs to be repaired by refreshing both two parts.

B. Enhanced Scheme based on Parity Check Bits on Generator Matrix

Based on the above analysis, SEUs on $\boldsymbol{G}$ are the main source for missing detection and false alarm and are difficult to be distinguished from those on *gflog*/*gfilog* tables. Considering the size of $\boldsymbol{G}$ is usually quite small in practice, we propose to perform separate fault detection for $\boldsymbol{G}$ based on parity check bits. For the $m \times k$ generator matrix $\boldsymbol{G}$, one parity bit is stored for each row based on the XOR operation, and then another check bit is stored for all the check bits. So totally m+1 parity bits are added. For each use of the generator matrix, the parity bit will be used to check the correctness of the elements in each row. If a mismatch is detected for any row, the parity bit of the row parity bits will be used to check whether the SEU happens on the parity bit of that row. The corrupted parity bit can be easily recovered by flipping, and the fault in the generator matrix can be recovered by refreshing the identified row. In addition, to avoid false alarm caused by SEUs on the recovered parity symbols, two copies of $\boldsymbol{p'_r}$ are stored as $\boldsymbol{p'_{r1}}$ and $\boldsymbol{p'_{r2}}$.

With the separate fault detection for $\boldsymbol{G}$ and $\boldsymbol{p'_r}$, the procedure of the fault detection and localization scheme is divided into 6 steps as shown in Fig.2, and the logic is shown in Fig. 3. Step ① is the check bits-based fault detection for generator matrix $\boldsymbol{G}$, and step ② is the first decoding with the received symbols. For step ③, the decoding results $\boldsymbol{d'}$ (including the received data symbols and the recovered erasure symbols) are re-encoded to generate new parity symbols ($\boldsymbol{p'_r}$), which are compared with the received ones ($\boldsymbol{p_r}$) for fault detection (④). If both copies of $\boldsymbol{p'_r}$ does not match $\boldsymbol{p_r}$, the decoding is repeated (⑤), and the new recovered data symbols $\boldsymbol{d''}$ are compared with that of the first decoding ($\boldsymbol{d'}$) for fault location between *gflog*/*gfilog* tables and intermediate variables (⑥). If faults are detection on generation matrix or *gflog*/*gfilog*, they would be repaired by refreshing. Otherwise, no repairing is needed.

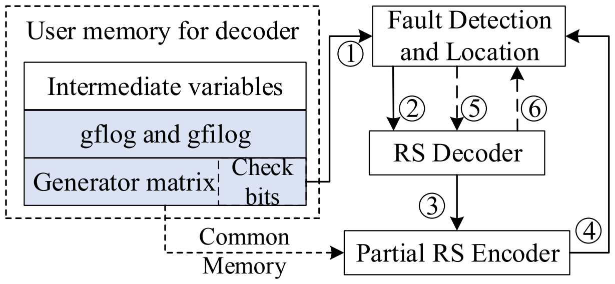

Fig.2. Structure of fault detection and location scheme

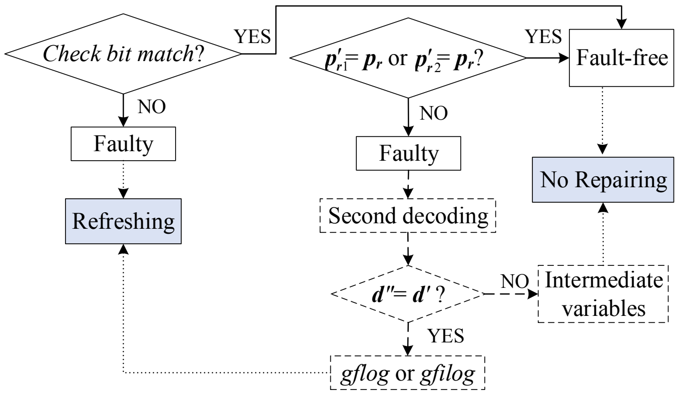

Fig. 3. Logic for fault detection and location

C. Performance Analysis of the Enhanced Fault Detection

In this subsection, the performance of the enhanced fault detection scheme is evaluated with the Missing Detection Probability (MDP) and the False Alarm Probability (FAP), respectively.

(1) MDP Analysis

The MDP is defined as the number of detected faults over the total number of faults that cause wrong decoding results. For the re-encoding-based fault detection, all the SEUs on intermediate variables and look-up tables can be detected when the number of erasures is less than m. But when the number of erasures E equals m, if the corrupted elements in *gflog* or *gfilog* are used in both the decoder and encoder, the faults may not be recognized (missing detection, MD). The MDP for SEUs on *gflog* and *gfilog*, and the average MDP for SEUs on all the user memories are analyzed as follows, respectively.

(a) MDP for SEUs on elements in *gflog*

For m erasures ($E = m$) with e data symbols, the total number of *gflog* look-ups that can cause decoding errors can be estimated as $N_{log}(m,e)$ according to (16). Meanwhile, *gflog* is used for $\boldsymbol{G'}\times\boldsymbol{d_e}$ in the decoder and $\boldsymbol{G_p}\times\boldsymbol{d'}$ in the encoder, and the common *gflog* lookups include the elements in $\boldsymbol{G'}$ ($N_{G'}(m,e)$ according to equation (14)) and the non-zero elements in $\boldsymbol{d_e}$ ($N_d = k-e$). SEUs on these $N_d + N_{G'}(m,e)$ elements will cause decoding errors and cannot be detected, so the MDP for SEUs on table *gflog* can be estimated as:

$$P_{\log}^{MD} = \sum_{e=1}^{m}(N_d + N_{G'}(m,e))P(m,e) \Big/ \sum_{e=1}^{m} N_{\log}(m,e)P(m,e), \quad (29)$$

in which $P(m,e)$ is calculated based on equation (11) with $m=E$.

(b) MDP for SEUs on elements in *gfilog*

For m erased symbols with e data symbols, the total number of different *gfilog* look-ups during the decoding and partial re-encoding are $N_{ilog}(m,e)$ (according to (23)) and $m_r\times k$, respectively, among which the number of common *gfilog* look-ups would be $N_3 = e\times(k-e)$ as defined in the third part of Section III.B. Then the MDP for SEUs on *gfilog* can be estimated as:

$$P_{\mathrm{ilog}}^{MD} = \sum_{e=1}^{m} N_3 P(m,e) \Big/ \sum_{e=1}^{m} N_{\mathrm{ilog}}(m,e)P(m,e) \quad (30)$$

(c) Average MDP for SEUs on all user memories

Since the missing detections are only caused by SEUs on *gflog* or *gfilog*, the average MDP for all the user memories can be estimated based on the fault tolerance probability (P_t^x) and memory size (S_x) for each part of the user memories as:

$$P_T^{MD} = \sum_{e=1}^{m}(N_d + N_{G'}(m,e) + N_3)P(m,e) \Big/ \sum_{x} S_x(1-P_t^x(m)) \quad (31)$$

(2) FAP Analysis

The 'false alarm' in this paper is defined as the event that an SEU happens and is detected, but the decoding result is correct. In other words, the fault is detected correctly, but the correct decoding result is mistaken to be wrong. In this case, 'un-necessary' recovery process will be performed for the current decoding, which will introduce 'un-necessary' delay. However, faults accumulation is eliminated by the recovery process, so the 'false alarm' is beneficial in the long run. For the partial re-encoding-based fault detection scheme, false alarm is only caused by the corrupted elements in *gflog* or *gfilog* that are only used during the re-encoding. The FAP for SEUs on these two tables and the average FAR for SEUs on all the user memories are analyzed as follows, respectively.

(a) FAP for SEUs on elements in *gflog*

For E erasures with e data symbols, the number of elements in *gflog* that would not cause decoding errors is $2^w-1-N_{log}(E,e)$. For partial re-encoding $\boldsymbol{G_p}\times\boldsymbol{d'}$, the *gflog* lookups that are different from those used in the decoding include the ones for m-E rows of $\boldsymbol{G}$ and e recovered data symbols. Since the elements in first column of $\boldsymbol{G}$ are all 1s, the number of new *gflog* look-ups for partial re-encoding would be $(m-E)(k-1)+e$. SEUs on these elements in *gflog* table would not affect the decoding results but would cause encoding errors, so the FAP for SEUs on *gflog* table can be estimated as:

$$P_{\log}^{FA} = \sum_{e=1}^{E}((m-E)(k-1)+e)P(E,e) \Big/ \sum_{e=1}^{E}(2^w-1-N_{\log}(E,e))P(E,e) \quad (32)$$

in which $P(E,e)$ is calculated based on equation (11).

(b) FAP for SEUs on elements in *gfilog*

Following a similar approach for FAP for *gflog*, the total number of elements in *gfilog* that would not cause decoding errors is $2^w-1-N_{ilog}(E,e)$, and the number of *gfilog* look-ups for partial re-encoding that are different from those used in decoding is $k(m-E)+e^2$. SEUs on these elements in *gfilog* table would not affect decoding results but would cause encoding errors, so the FAP for SEUs on *gfilog* can be estimated as:

$$P_{\mathrm{ilog}}^{FA} = \sum_{e=1}^{E}(k(m-E)+e^2)P(E,e) \Big/ \sum_{e=1}^{E}(2^w-1-N_{\mathrm{ilog}}(E,e))P(E,e) \quad (33)$$

(c) Average FAP for SEUs on all user memories

Since the false alarms are only caused by SEUs on *gflog* or *gfilog*, the average FAP for all the user memories can be estimated based on the fault tolerance probability (P_t^x) and memory size (S_x) for each part of the user memories as:

$$P_{avg}^{FA} = \sum_{e=1}^{E}(e^2+e+(m-E)(2k-1))P(E,e) \Big/ \sum_{x} S_x P_t^x(E) \quad (34)$$

V. Evaluation of the Proposed Fault Tolerant RS Decoder

This section presents the fault injection experiments and evaluates the proposed fault detection and location scheme for SEUs on the user memory in the RS-EC decoder.

A. Resource and Time Overhead

The decoder for (6, 3) RS-EC with the proposed fault detection and localization scheme was implemented using Verilog and mapped on the same target device, a Xilinx Zynq 7000 SoC (xc7z030ffg 676-1). To improve the resource efficiency, the partial re-encoding ($\boldsymbol{G_p}\times\boldsymbol{d'}$) reuses the module for $\boldsymbol{G'}\times\boldsymbol{d_e}$ in the decoder, and the two copies of the newly generated parity symbols ($\boldsymbol{p'_{r1}}$ and $\boldsymbol{p'_{r2}}$) are stored in $\boldsymbol{d_e}$ considering that k is usually larger than $2m$. The usage of LUTs, registers and BRAMs of the original RS-EC decoder and that with the proposed fault detection and location scheme are listed in Table V. As we can see, the proposed RS-EC decoder consumes additional 73 LUTs and 50 registers (for partial re-encoding and parity bit check for generation matrix G), thus the resource overhead is about 1.03 times of that of the original decoder. The decoding time in cycles is compared in Table VI. The time overhead of the proposed solution is about 1.16 and 2.17 times of that of the original decoder for the cases without and with fault, respectively.

TABLE V. Resource Usage of Unprotected and Proposed RS Decoder

	LUTs	Registers	BRAMs
Unprotected	2458	1238	1
Proposed	2518 (1.02x)	1311(1.04x)	1

TABLE VI. TIME OF UNPROTECTED AND PROPOSED RS DECODER IN CYCLES

Erasures		1	2	3
Unprotected		178	187	204
Proposed	Free	206 (1.16x)	216 (1.15x)	232 (1.14x)
	Fault	387 (2.17x)	407 (2.17x)	439 (2.15x)

B. Platform for SEUs Injection Experiments

To assess the effectiveness of the proposed scheme to detect and locate faults in the user memory of RS decoder, SEUs have been injected using an adapted version of the fault injection tool in [30] and implemented on a Zedboard. The experimental setup for injection on the unprotected and proposed RS decoder with fault detection and location is shown in Fig. 5. The injection platform consists of the Processing System (PS) and the Programmable Logic (PL). The PS consists of the ARM Cortex-A9 processor and dedicated controllers for different peripherals e.g. DDR memory controller, SD controller, UART controller etc. The DDR controller is responsible for storing the list of bits in the registers and the BRAMs. The list is generated during the compile time in Vivado and is used by the injection algorithm for reliability evaluation of the original decoder and the performance evaluation of the fault detection and location scheme. The UART module in the PS is responsible for logging results to the PC. The PL part consists of two copies of the RS-EC decoder i.e. Golden and Design Under Test (DUT). Moreover, a comparator and a synchronizer block are also present in the PL part. These modules are responsible for concurrent error detection in the Golden and DUT RS-EC decoders, and the mismatches are recorded by ARM. The synchronizer module is responsible for controlling the clock to DUT and Golden copies of the RS-EC decoder. Furthermore, the RS-EC decoder receives inputs from the ARM processor in the PS part, so the test patterns are totally software-controlled. Another important module housed by the PL part is the Internal Configuration Access Port (ICAP) module, which allows the ARM processor to access the user memory related to the DUT decoder in run-time for error injection. The modules in the PL region are connected to the PS region through AXI buses.

The ARM processor runs the software that controls the fault injection process. The fault injection starts by freezing the clock to the RS-EC decoder in the PL part (through the synchronizer module). This is followed by reading back the target bit from a register/BRAM through the ICAP port. The address of the user memory bits for fault injection is extracted from the fault list stored in DDR memory. The read back values are corrupted by inserting a bit flip for SEU emulation and written back. This is followed by resuming the design's clock. For reliability evaluation of original RS-EC decoder, the mismatches between the Golden and DUT decoders are collected by ARM and communicated to PC through UART interface. For the performance evaluation of the proposed scheme, the fault detection and location results are also collected.

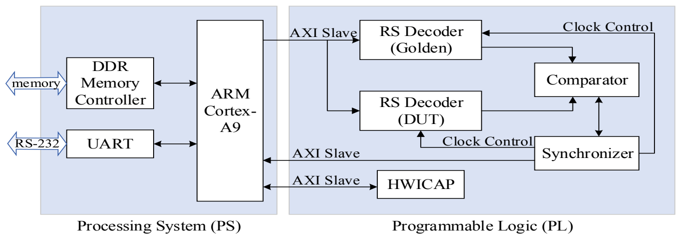

Fig. 5. Fault injection platform of RS Decoder

C. Performance Evaluation of Proposed Fault Detection and Location Scheme for RS-EC Decoder

The MDP and FAP of the proposed fault detection scheme are tested based on the fault injection experiments for each part of the user memory, and the accuracy of the fault location scheme is also evaluated correspondingly.

Similar to the reliability evaluation of original decoder, 100 RS-EC codewords are generated randomly for the fault injection experiments on each bit for each case of $E = 1, 2$, or 3. To speed up the experiments, SEUs are only injected during the effective period of each part of memories, so the total number of SEU injections on user memory x is $N_I^x = 100 \times S_x \times 8$ for each case of E, where S_x is the number of words for memory x. Then the number of SEUs that cause decoding errors (on critical bits) are recorded as N_{err}^x, among which the missing detections are counted as N_{MD}^x. For the SEUs that do not cause decoding errors ($N_{free}^x = N_I^x - N_{err}^x$), the number of false alarms is recorded as N_{FA}^x. Finally, we can get the total number of SEU injections as $N_I^T = \sum_x N_I^x$, the total number of SEUs causing decoding errors as $N_{err}^T = \sum_x N_{err}^x$, and the total number of missing detections as $N_{MD}^T = \sum_x N_{MD}^x$.

(1) Testing results of MDP

The total MDP is measured as $P_T^{MD} = N_{MD}^T / N_{err}^T$ for $E = 1$, 2 or 3, respectively, and the MDP for user memory x is measured as $P_x^{MD} = N_{MD}^x / N_{err}^x$. The testing results for total MDP are listed in Table VI, and the theoretical estimates are also provided for comparison. As expected, for $E = 1$ or 2, all the SEUs that will cause decoding errors are detected ($P_T^{MD} = 0$). For $E = 3$, we have P_T^{MD}=13.07%, and further analysis shows that all the missing detections are caused by the SEUs on *gflog* and *gfilog* tables. As shown in Table VII, about 46% and 39% of the SEUs on critical bits of *gflog* or *gfilog* tables cannot be detected by the proposed scheme, respectively. This means that large portion of the critical bits in the *gflog* or *gfilog* tables are used in both the decoder and the partial encoder when the number of erasures achieves the recovery limit of RS-EC. Assuming all the detected faults can be recovered by refreshing and re-decoding, the reliability of the RS-EC decoder can be further improved from 96.47%, 94.65% and 91.22% (in Table III) to 100%, 100% and 98.9% (1-(1-91.22%)×13.07%) for E = 1, 2 or 3, respectively, which indicates the effectiveness of the fault detection scheme. In addition, the theoretical estimates are very close to the experiment results in Table VI and Table VII, which verify the correctness of the analysis approach.

TABLE VI. MISSING DETECTION PROBABILITY FOR ALL USER MEMORIES

	N_I^T	N_{err}^T	N_{MD}^T	P_T^{MD}(Expe)	P_T^{MD}(Theory)
$E = 1$	508800	34191	0	0	0
$E = 2$	508800	53271	0	0	0
$E = 3$	508800	84766	11077	13.07%	15.13%

TABLE VII. MISSING DETECTION PROBABILITY FOR GFLOG/GFILOG (E=3)

	N_I	N_{err}	N_{MD}	P_x^{MD}(Expe)	P_x^{MD}(Theory)
gflog	204000	12821	5936	46.30%	45.90%
gfilog	204000	13476	5141	38.15%	39.01%

(2) Testing results of FAP

FAP is measured as $P_x^{FA} = N_{FA}^x / N_{free}^x$ for *gflog* and *gfilog*, respectively, and the results are listed in Table VIII, in which the theoretical estimates are also provided for comparison. Based on the testing results, the FAP for *gflog* and *gfilog* are about 3.2% and 4.1%, respectively, for single erasure, and decreases to 1% and 1.65%, respectively, for 3 erasures. This is expected because more elements in these tables are used for decoding for more erasures, then the ones only used in re-encoding becomes fewer. If weighted by the memory size, the FAP for all the memories in the decoder can be calculated based on equation (34), and the results are listed in Table IX. Since *gflog* and *gfilog* dominates the user memory of the decoder, the average FAP is just slightly smaller and the trend is similar for different erasures. Again, the theoretical estimates are also close to the testing results.

TABLE VIII. FALSE ALARM PROBABILITY FOR GFLOG AND GFILOG

Number erasures	1		2		3	
	Theory	Expe	Theory	Expe	Theory	Expe
gflog	4.45%	**3.19%**	2.67%	2.10%	0.86%	**1.01%**
gfilog	5.22%	**4.11%**	3.42%	2.64%	1.93%	**1.65%**

TABLE IX. AVERAGE FALSE ALARM PROBABILITY FOR ALL USER MEMORY

	E=1	E=2	E=3
Theoretical	4.05%	2.62%	1.25%
Experimental	3.06%	2.04%	1.19%

(3) Fault location accuracy

The fault location accuracy is measured by the ratio of the number of correctly located faults over the number of detected faults. The evaluation is performed for each part of the memory. As expected, all the faults can be located correctly, which proves the completeness of the proposed scheme.

VI. CONCLUSIONS

In this paper, the reliability on the user memory of RS-EC decoder implemented on FPGAs is first studied by theoretical analysis and fault injection experiments, and the results show that the decoder itself has strong fault tolerance against SEUs. Then a fault detection scheme based on partial re-encoding and a fault localization scheme based on re-decoding are proposed to protect the RS decoder. To further enhance the performance of the basic fault detection and localization scheme, generator matrix is protected separately by parity check bits, and the probability of missing detection and false alarm of the final scheme is analyzed. Fault injection experiments show that 1) the fault detection scheme could detect all the faults if the number of erasures is less than m, and missing detections appear for m erasures with a very small probability; 2) false alarms exist with a small probability, but they only introduce a decoding overhead and are also beneficial to avoid faults accumulation. 3) the fault location scheme could locate all the detected faults on intermediate variables, generator matrix and two look-up tables, with accuracy of 100%.

REFERENCES

[1] R. E. Blahut. Theory and Practice of Error Control Codes. Addison Wesley, 1984.

[2] T. Zhang and K. K. Parhi. "On the high-speed VLSI implementation of errors-and-erasures correcting reed-solomon decoders". In Proceedings of the 12th ACM Great Lakes symposium on VLSI, New York, USA, 2002.

[3] F. Hernando, K. Marshall, M E. O'Sullivan . "The Dimension of Subcode-Subfields of Shortened Generalized Reed Solomon Codes", Designs Codes and Cryptography, vol. 69, pp. 131–142 , 2013.

[4] J. Li. "The efficient implementation of Reed-Solomon high rate erasure resilient codes", in Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 1097-1100, 2005.

[5] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," ACM Computer Communication review, vol. 27, pp. 24–36, Apr. 1997.

[6] A. Al-Shaikhi and J. Ilow, "Packet Loss Recovery Codes Based on Vandermonde Matrices and Shift Operator", in Proc. IEEE International Symposium on Information Theory, pp. 1058-1062, 2008.

[7] G.C. Cardarilli, A. Leandri, P. Marinucci, M. Ottavi, S. Pontarelli, M. Re, A. Salsano, "Design of a fault tolerant solid state mass memory, IEEE Transactions on Reliability", vol. 52, pp. 476 – 491, 1999.

[8] J. S. Plank. "A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems". Software Practice & Experience, vol. 27, no. 9, pp. 995-1012, 1996.

[9] A. Fikes, Colossus, Successor to Google File System, Available Online: http://google.com/en/us/university/relations/facultysummit2010/storage_architecture_and_challenges.pdf

[10] S. Muralidhar et al., "F4: Facebook's warm BLOB storage system," 11th ACM/USENIX Symp. Oper. Syst. Design Implement. (OSDI), 2014.

[11] J. Foust, "SpaceX's space-Internet woes", IEEE Spectrum, vol. 56, no. 1, Jan. 2019.

[12] M. Harris, "Tech giants race to build orbital internet", IEEE Spectrum, vol. 55, no. 6, June 2018.

[13] D. Li, X. Shen, N. Chen, and Z. Xiao, "Space-based information service in Internet Plus Era", Science China Information Sciences, vol.60, 2017.

[14] C. Fei, B. Zhao, W. Yu, and C. Wu, "Towards Efficient Data Collection in Space-Based Internet of Things", Sensors, vol. 19, 2019.

[15] E.G. Stassinopoulos and J.P. Raymond, "The space radiation environment for electronics", Proceedings of the IEEE, vol. 76, no. 11, Nov. 1988.

[16] M. Tali, R. G. Alía, M. Brugger, et al. "High-Energy Electron-Induced SEUs and Jovian Environment Impact", IEEE Transactions on Nuclear Science, vol. 64, no. 8, Aug. 2017.

[17] S. López, "The Promise of Reconfigurable Computing for Hyperspectral Imaging Onboard Systems: A Review and Trends", Proceedings of IEEE, March 2013.

[18] F. Siegle, T.Vladimirova, J. Ilstad, et al. "Availability analysis for satellite data processing systems based on SRAM FPGAs", IEEE Transactions on Aerospace Electronic Systems, vol. 52, no. 3, pp. 977-989, 2016.

[19] F. L. Kastensmidt, L. Carro and R. Reis, "Fault-tolerance Techniques for SRAM-based FPGAs", New Haven, Springer 2006.

[20] Z. Gao, L. Yan, J. Zhu, R. Han, et al. "Radiation tolerant viterbi decoders for on-board processing (OBP) in satellite communications," in China Communications, vol. 17, no. 1, pp. 140-150, Jan. 2020.

[21] M.K. Jaswal, D. Mallik, M. Kaur, "Radiation hardened SEU tolerant Reed Solomon encoder and decoder", in 3rd International Conference on Signal Processing and Integrated Networks, SPIN 2016.

[22] G. C. Cardarilli, S. Pontarelli, M. Re, A. Salsano. "Concurrent error detection in reed–solomon encoders and decoders". IEEE Transactions on Very Large Scale Integration Systems, vol. 15, pp. 842-846, 2007.

[23] S. Pontarelli, L. Sterpone, G.C. Cardarilli, et al. "Self Checking Circuit Optimization by means of Fault Injection Analysis: A Case Study on Reed Solomon Decoders", in Proc. IEEE International On-line Testing Symposium, pp. 194-196, 2007.

[24] J. K. Omura and J. L. Massey, "Computational method and apparatus for finite field arithmetic," U.S. Patent 4,587,627, May 6, 1986

[25] J. Lacan and J. Fimes, "Systematic MDS erasure codes based on Vandermonde matrices," IEEE Commun. Lett., vol. 8, no. 9, pp. 570–572, Sep. 2004.

[26] M.S. Feali, A. Ahmadi, A. Hamidi, M. Ahmadi, "Fixed-point arithmetic error analysis of sparse LU decomposition on FPGAs", in Proc. International Symposium on Signals, Circuits and Systems, 2017.

[27] G. Wu, Y. Dou, J. Sun, and G. D. Peterson, "A High Performance and Memory Efficient LU Decomposer on FPGAs," IEEE Transactions on Computers, vol. 61, no. 3, pp. 366-378, 2012.

[28] M. K. Jaisval and N. Chandrachoodan, "FPGA based high performance and scalable block LU decomposition architecture" IEEE Transactions on Computers, vol.61, no. 1, pp.60-72, 2012.

[29] H. P. Oquendo, P. S. Pacheco, "Bounds for the 1-norm of the inverses of some triangular matrices", Linear Algebra and its Applications, vol. 495, pp. 163-173, 2016.

[30] A. Ullah, P. Reviriego, J. A. Maestro, "An Efficient Methodology for On-Chip SEU Injection in Flip-Flops for Xilinx FPGAs," in *IEEE Transactions on Nuclear Science*, vol. 65, no. 4, pp. 989-996, April 2018.