A CRC-Aided LDPC Erasure Decoding Algorithm for SEUs Correcting in Small Satellites

Hao Zheng[1(✉)], Zinan Song[1], Shuyi Zhang[1], Shuo Chai[1], and Liwei Shao[2]

[1] School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China
3120130330@bit.edu.cn
[2] Research Institute of BIT in Zhongshan, Zhongshan 528400, China

Abstract. Bit-flip caused by SEUs is one of the main reasons resulting in small satellites malfunction. This paper proposes a LDPC erasure decoding method aided by CRC to detect and correct errors in stored data. The key idea is that the encoded message is divided into multiple fragments protected by individual CRC so that fragments with error could be detected and corrected. Moreover, CRC is also used as an early stop criterion of decoding. Simulation and implementation results show that the proposed method has better performance compared with MS decoding both in error correcting and hardware requirements.

Keywords: LDPC erasure codes · Decoding · CRC · SEU

1 Introduction

Small satellites that could be launched in short time-scales and tight budgets have provided fast and cheap accesses to space [1]. These low-cost satellites make it possible for commercial organizations and emerging space nations to conduct independent low earth orbits (LEO) missions. Therefore, there is a growing need of small satellites.

In small satellites applications, stored digital data in random-access memory (RAM) or flash is easily suffered from single-event upsets (SEUs) caused by radiation which is impossible to be shielded [2]. More than half of system malfunctions in satellites are the result of bit-flips in stored data caused by SEUs [3].

To secure the data, many methods have been proposed, which could be divided in to three categories: (1) Multiple modular redundancy methods, such as those proposed by P.K. Samudrala in [4] and S. Hao in [5]. Methods in this category take advantages of making multiple copies of data so that voting logic can be used to detect and correct errors. However, these schemes can only offer protection against the effect of bit-flips when less than a half of data copies are affected and require larger RAM or flash. (2) Error-Correcting code based methods, such as RS codes used by Y. Zhang in [6] and G.C. Cardarilli in [7], LDPC codes used by B. Vasic in [8]. These methods can provide excellent correcting performance against bit-flips, but their decoders are usually complicated to implement. (3) Mixed methods, such as Hsiao codes and triple

X.-L. Huang (Ed.): MLICOM 2016, LNICST 183, pp. 35–43, 2017.
DOI: 10.1007/978-3-319-52730-7_4

modular redundancy method proposed by Y. Zhao in [9], which in fact are compromise between implementation complexity and error correcting performance. Yet aforementioned methods either require multiple storage space or high complexity.

In this paper, we propose a new method using CRC codes and low-density parity-check (LDPC) erasure codes to detect and correct data errors caused by SEUs. This method applies CRC codes to detect errors in each data fragments that are divided from original data segments encoded by LDPC erasure codes and terminate the decoding process early, while the correcting step of the method is more accurate and requires less on hardware.

The rest of this paper is organized as follow. Section 2 will briefly introduce the LDPC erasure codes adopted and describe the system. Section 3 will gives the CRC-aided LDPC erasure decoding algorithm and discuss the CRC selection problem. Simulation and implementation results are presented in Sect. 4, and Sect. 5 concludes this paper.

2 The LDPC Erasure Codes and System Description

2.1 LDPC Erasure Codes and Encoding

LDPC erasure codes are a kind of concatenated code with low-density parity-check matrix, and its outer code and inner code are random LDPC code and irregular-repeat-accumulate (IRA) LDPC code, respectively.

The parity-check matrix of LDPC erasure codes with code length n_i and information bits length k_o is described below

$$\begin{cases} H = \begin{bmatrix} H_o & 0 \\ H_{i,u} & H_{i,p} \end{bmatrix} \\ H_o = \left[H_{o,u} \middle| H_{o,p} \right] \end{cases} \tag{1}$$

where H_o is a $(n_o - k_o) \times n_o$ parity-check matrix of outer code with code length n_o, $H_{i,u}$ and $H_{o,u}$ are binary random matrices with size $(n_o - k_o) \times k_o$ and $(n_i - k_i) \times k_i$, $H_{o,p}$ and $H_{i,p}$ are $(n_o - k_o) \times (n_o - k_o)$ and $(n_i - k_i) \times (n_i - k_i)$ matrices in *dual-diagonal* pattern

$$H_{*,p} = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{2}$$

where * is o or p. Moreover, the parity-check matrix of inner code is $H_i = \left[H_{i,u} \middle| H_{i,p} \right]$, whose size is $(n_i - k_i) \times n_i$, $k_i \geq n_o$. As $H_{o,p}$ and $H_{i,p}$ are both in *dual-diagonal* pattern, both H_o and H_i could easily be transformed into systematic form, from which generator

matrices of both outer code and inner code, denoted as G_o and G_i, can be obtained. With G_o and G_i, codewords c could be gained through following steps

Step 1. Gain the outer codeword $c_o = m \cdot G_o$.

Step 2. Gain the inner codeword, namely the codeword of LDPC erasure codes, c. If $k_i = n_o$, then $c = c_o \cdot G_i$. Otherwise, fill zeroes at the end of c_o to make it length equal to n_o, and let $c = c_o \cdot G_i$.

2.2 System Description

Figure 1 depicts the procedure that data input into and output from RAM/Flash. The input data m is encoded by LDPC erasure encoder and become a codeword c so that errors can be corrected at output stage, then c can be decomposed into different data fragments p, each of which can be protected by individual CRC, and written into RAM/flash. When stored data need to be read, data fragments p' are read out and checks by CRCs, and data fragments with errors will be erased. Then c' can be composed by various data fragments, and all erased data will be recovered by LDPC erasure decoder and get the output data m'.

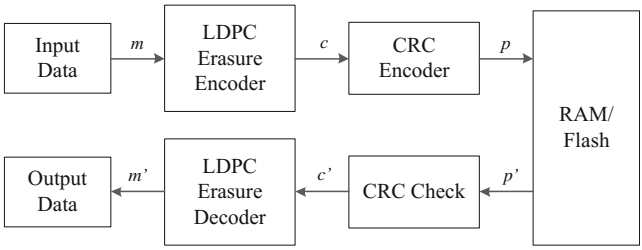

Fig. 1. In the write-in part (top): the input data is encoded by LDPC Erasure Encoder, then decomposed into different fragments protected by individual CRC. In the read-out part (button): the read-out packages that do not pass CRC check will be erased and recovered by LDPC erasure decoder.

Figure 2 illustrates the decomposition of LDPC erasure codewords and multiple data fragments that protected by individual CRC. The original data segment is decomposed into l fragments protected by individual CRC respectively, while the parity segment is decomposed into f fragments. Therefore, length of all fragments is $n_i + (l+f)k_{CRC}$ bits, in which length of data segment and parity segment is $k_o + lk_{CRC}$ bits and $n_i - k_o + fk_{CRC}$ bits respectively.

In this paper, CRCs in p will be used in three ways below. The first is to detect errors in fragments so that the decision that whether the decoding procedure is necessary could be made; the second is to decide which fragments should be erased or used as correct information in the decoding procedure; the last is to early terminate the decoding process by checking whether all data fragments could satisfy CRCs. If a

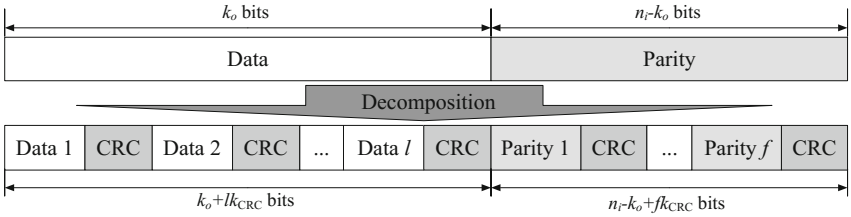

Fig. 2. Decompose of LDPC erasure codewords (top) into multiple fragments (button) protected by individual CRC.

fragment is decided as correct one by its CRC, it is assumed to be correct despite the undetected error of its CRC. Then, when the decoder is initialized, bits inside this fragment will be treated as correct bits. Due to the small probability that bit-flips caused by SEUs happened more than once in such a small fragment of data, the probability of undetected error is sufficiently small.

3 LDPC Erasure Decoding Based on CRCs

3.1 CRC-Aided LDPC Erasure Decoding

The proposed CRC-Aided LDPC erasure decoding algorithm is based on the peeling decoding (PD) in [10]. To describe PD, an LDPC erasure code should be represented as a Tanner graph [11, 12], which is constituted by a set of variable nodes connected to a set of check nodes. The PD is initialized by removing all the variable nodes whose corresponding bits are not erased and complementing the parity of check nodes connected to removed nodes whose value are 1. Then at each iteration, the PD looks for degree-one check nodes and gives their parity value to connected variable nodes, and removes those variable nodes while reverses the parity of check nodes connected to them if their value is 1. The PD will stop decoding if a codeword is obtained (i.e. decoding successes) or there is no more check nodes with one degree (i.e. decoding fails).

To describe the proposed algorithm, some notations should be made. The parity-check matrix of LDPC erasure codes is denoted as H, and described as

$$H = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,n_i-1} & h_{1,n_i} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,n_i-1} & h_{2,n_i} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{m-1,1} & h_{m-1,2} & \cdots & h_{m-1,n_i-1} & h_{m-1,n_i} \\ h_{m,1} & h_{m,2} & \cdots & h_{m,n_i-1} & h_{m,n_i} \end{bmatrix} \tag{3}$$

where $h_{r,q} \in \{0, 1\}$ is the elements in H, in which $r = 1, 2, \ldots, m$ and $q = 1, 2, \ldots, n_i$, and $m = n_i - k_o$ is the number of rows in H. Let the decoded codeword and the number of erased bits in each parity-check equation to be $\hat{c} = \{\hat{c}_q\}$ for $q = 1, 2, \ldots, n_i$ and $C_{eb} = \left\{C_{eb}^r\right\}$ for $r = 1, 2, \ldots, m$, respectively.

With these notations, the CRC-Aided LDPC erasure decoding algorithm is described in the following steps, where the major contribution of this paper is Step 4.

Step 1. Initialization. Initialize the decoded codeword $\hat{c} = \{\hat{c}_q\}$ for $q = 1, 2, \ldots, n_i$ using CRC check results

$$\begin{cases} \hat{c}_q = 0, & \text{if } p'_q \text{ is erased} \\ \hat{c}_q = 1 - 2p'_q, & \text{if } p'_q \text{ is correct} \end{cases}, \; p'_q \in \{0, 1\}, q = 1, 2, \ldots, n_i, \quad (4)$$

and set the erased bits counter C_e to the number of bits that are erased, namely the number of zeroes in $\hat{c}$

$$C_e = n_i - \sum_{q=1}^{n_i} \left|\hat{c}_q\right|, \quad (5)$$

count the number of erased bits in each parity-check equation

$$C_{eb}^r = \sum_{\substack{ch_{r,q}=1 \\ \hat{c}_q = 0}} h_{r,q}, \text{ if } \prod_{h_{r,q}=1} \hat{c}_q \neq 1, r = 1, 2, \ldots, m, q = 1, 2, \ldots, n_i. \quad (6)$$

Step 2. Correct erased bits. Search in C_{eb} and find r that let $C_{eb}^r = 1$. Update the only erased bits in this parity-check equation

$$\hat{c}_q = \prod_{\substack{h_{r,q'}=1 \\ q' \neq q}} \hat{c}_{q'}, C_{eb}^r = 1, r = 1, 2, \ldots, m, q = 1, 2, \ldots, n_i. \quad (7)$$

Step 3. Update C_e and C_{eb} with (5) and (6), respectively.

Step 4. Stop decoding. If any condition below is satisfied, the decoding process will terminate and declares a success decoding.
(1) $C_e = 0$;
(2) All fragments in data segment can pass CRCs check.

If any condition below is satisfied, the decoding process will terminate and declares a fail decoding.

(1) $C_e \neq 0$, while $C_{eb}^r \neq 1$, for any $r = 1, 2, \ldots, m$;
(2) The number of iterations reaches the max iteration number.

Otherwise, the decoding process will continue and add the number of iterations by 1.

It is noted that the decoded codeword $\hat{c}$ is also an updatable indicator that indicates bits that been erased, when a bits is erased due to CRCs its corresponding $\hat{c}_q$ is 0. Moreover, compared with conventional PD, decoders using the proposed algorithm can employ parallel structure to reduce decoding delay.

3.2 Selection of CRC

The proposed CRC-Aided LDPC erasure decoding algorithm is based on the CRCs, and the performance of it relies on the error detection capability of CRC. Thus, selecting efficient CRC is important. Normally, the undetected error probability P_{ud} is used to evaluate the performance of CRC [13], which is given by

$$P_{ud} = \sum_{j=1}^{N} A_j p^j (1-p)^{N-j} \tag{8}$$

where N is the length of data that CRC protects, p is the bit error probability, the set $\{A_j\}$ is the weight distribution of generator polynomial $g(x)$, while R is the order of $g(x)$. To find the optimal CRC code, there is the tremendous code space need to be searched, which is not an easy work. However, [13] has reported many efficient and proper CRC codes. From [2], the bit-flips caused by SEUs is at most 4 times per day every 1 Mb, in other words, the bit error probability is less than 10^{-10} such that P_{ud} is even small. Hence, the CRC code used could be selected from [13].

4 Simulation and Implementation Results

Simulations of the CRC-Aided LDPC erasure decoding algorithm regard the code error rate (CER) and average number of iterations (ANI) to verify error performance and decoding speed of it. In all experiments, the erasure channel model describe in [14] is employed, and the maximum number of iteration and codewords are 100 and 10^6. The employed LDPC erasure codes and proper CRC codes are described in the CCSDS standard [15] and [13], respectively.

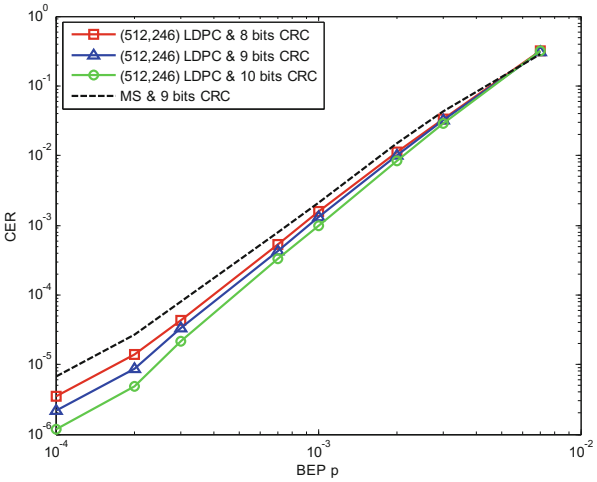

Fig. 3. Codeword error rate (CER) versus bit error probability (BEP) p.

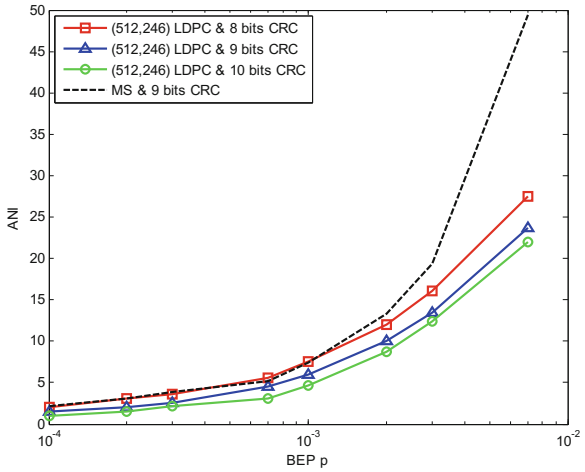

Fig. 4. Average number of iterations (ANI) versus bit error probability (BEP) p.

To confirm that the proposed method has a better performance for correcting storage errors, different configurations have been applied under the same code rate and compared with Min-Sum (MS) Algorithm. Figures 3 and 4 indicate the CER performance and ANI of proposed algorithm, which show that the proposed algorithm has outperformed in both CER and ANI compare to MS. However, CRC codes with different lengths effect the performance of proposed algorithm, longer CRC codes owns better performance, as longer CRC codes owns lower undetected error probability. But longer CRC codes require extra storage space and hardware. Hence, to meet the requirements, multiple simulations, like CER and ANI performance, should be conducted when proposed method is employed.

Table 1. The implementation of proposed algorithm on FPGA

Logic utilization	Proposed	MS
Slices	148	576
LUTs	180	436
Block RAMs	2	4
Max clock	357.3 MHz	280.6 MHz

In order to evaluate the complexity of proposed method, both proposed methods and MS algorithm using (512,246) LDPC aided by 9 bits CRC has been implemented on a Xilinx Virtex5 FPGA. Table 1 shows the implementation results of both methods, from where it is can be observed that the proposed algorithm require less on hardware because of its correcting step could achieved using one-bit addition operations, instead of several multi-bits operations. Thus, the proposed method outperforms MS both in error correcting and hardware requirements.

5 Conclusions

Using CRC codes to detect errors in stored data, a new method is proposed using LDPC erasure codes correct data errors caused by SEUs. In this method, message encoded by LDPC erasure codes are decomposed into several fragments, each of which are protected by individual CRC, so that fragments fail to pass CRC check can be detected and then corrected by LDPC erasure decoding process. Due to the accuracy and implementing fitness of its correcting step, the proposed algorithm is more efficient and requires less on hardware compare with MS algorithm with CRCs.

Acknowledgments. This research is supported by Opening Fund Project of Space Target Measurement Key Laboratory of PLA General Armament Department and Guangdong Provincial Science and Technology Project (No. 2015B010101002).

References

1. Wicks, A., da Silva Curiel, J., Ward, M.: Fouquet: advancing small satellite earth observation: operational spacecraft, planned missions and future concepts. In: 14th Annual AIIA/USU Conference Small Satellites, pp. 21–24. Logan (2000)
2. Ma, R., Zhang, Y., Bai, Z.: Practice V Satellite and its flight achievements. Aerosp. Chin. **11**, 5–10 (1999)
3. Wang, C.: The influene with reliability of motional satellite by the single-event phenomena. Semicond. Inf. **35**, 1–8 (1998)
4. Samudrala, P.K., Ramos, J., Katkoori, S.: Selective triple modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs. IEEE Trans. Nucl. Sci. **51**(5), 2957–2969 (2004)
5. Hao, S., Yang, Z., Chai, Z.: Study on memory's fault tolerant design based on multiple module redundancy reconfiguration. Comput. Meas. Control **17**, 190–194 (2009)
6. Zhang, Y., Yang, G., Li, H., Chang, L.: Parallel reed-solomon error correction for spaceborne mass memory system. Aerosp. Control **3**, 86–89 (2009)
7. Cardarilli, G.C., Leandri, A., Marinucci, P., Ottavi, M., Pontarelli, S., Re, M., Salsano, A.: Design of a fault tolerant solid state mass memory. IEEE Trans. Reliab. **52**(4), 476–491 (2003)
8. Vasic, B., Ivanis, P., Brkic, S.: Low complexity memory architectures based on LDPC codes: benefits and disadvantages. In: 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), pp. 11–18. IEEE Press, Serbia (2015)
9. Zhao, Y., Hua, G.: Method of fault tolerant design for memory. Aerosp. Control Appl. **35**(3), 61–64 (2009)
10. Richardson, T., Urbanke, R.: Peeling decoder and order of limits. In: Richardson, T., Urbanke, R. (eds.) Modern Coding Theory, pp. 115–122. Cambridge University Press, Cambridge (2008)
11. Tanner, R.M.: A recursive approach to low complexity codes. IEEE Trans. Inf. Theory **27**(5), 533–547 (1981)
12. Loeliger, H.A.: An introduction to factor graphs. IEEE Signal Process. Mag. **21**(1), 28–41 (2004)

13. Chun, D., Wolf, J.K.: Special hardware for computing the probability of undetected error for certain binary CRC codes and test results. IEEE Trans. Commun. **42**(10), 2769–2772 (1994)
14. Xia, H., Cruz, J.R.: On the performance of soft Reed-Solomon decoding for magnetic recording channels with erasures. IEEE Trans. Magn. **39**(5), 2576–2578 (2003)
15. CCSDS Orange Book: Erasure Correcting Codes for Use in Near-Earth and Deep-Space Communications. http://public.ccsds.org/publications/archive/131x5o1.pdf