Research Article

Design and Analysis of Adaptive Message Coding on LDPC Decoder with Faulty Storage

Guangjun Ge^{1,2} and Liuguo Yin^{ID 2,3}

¹School of Aerospace, Tsinghua University, Beijing, China

²EDA Laboratory, Research Institute of Tsinghua University in Shenzhen, Shenzhen, China

³School of Information Science and Technology, Tsinghua University, Beijing, China

Correspondence should be addressed to Liuguo Yin; yinlg@tsinghua.edu.cn

Received 23 November 2017; Revised 9 February 2018; Accepted 19 February 2018; Published 22 March 2018

Academic Editor: Qin Huang

Copyright © 2018 Guangjun Ge and Liuguo Yin. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Unreliable message storage severely degrades the performance of LDPC decoders. This paper discusses the impacts of message errors on LDPC decoders and schemes improving the robustness. Firstly, we develop a discrete density evolution analysis for faulty LDPC decoders, which indicates that protecting the sign bits of messages is effective enough for finite-precision LDPC decoders. Secondly, we analyze the effects of quantization precision loss for static sign bit protection and propose an embedded dynamic coding scheme by adaptively employing the least significant bits (LSBs) to protect the sign bits. Thirdly, we give a construction of Hamming product code for the adaptive coding and present low complexity decoding algorithms. Theoretic analysis indicates that the proposed scheme outperforms traditional triple modular redundancy (TMR) scheme in decoding both threshold and residual errors, while Monte Carlo simulations show that the performance loss is less than 0.2 dB when the storage error probability varies from 10^{-3} to 10^{-4} .

1. Introduction

Low-Density Parity-Check (LDPC) codes are widely used in space communications due to their capacity-approaching capabilities [1]. The outstanding performance of LDPC is based on the soft-decoding algorithms [2] which consume a large number of memories. However, the radiation environment will give rise to fault problems for memories when LDPC decoders are used in the spacecraft [3]. Such unreliable storage will severely degrade the performance of LDPC codes. Thus, it is important to consider the robustness of LDPC decoders utilizing unreliable memories.

There are studies on the effects of unreliable hardware on LDPC decoders. Varshney considered the thresholds and residual errors of LDPC codes with the faulty Gallager A decoding in the earlier stage [4]. Extended studies on faulty Gallager B decoders were then developed in [5–7]. Besides these bit flipping decoding algorithms, the belief propagation (BP) decoding of LDPC on noisy hardware was studied in [8, 9], where infinite-precision message with additive Gaussian noise was considered. Finite-precision message for

the min-sum decoding of LDPC was studied in [10–12]. It showed that quantizing messages with more bits was not always beneficial for LDPC decoders with hardware errors.

In general, the existing works treated each finite-precision message as an integer, while this paper discusses the various impacts of different bits of the finite-precision message. We develop a discrete density evolution analysis for LDPC decoders with faulty messages. It indicates that the sign bits of the messages play the most important role in the decoding performance of LDPC codes, which means setting protection on sign bits is efficient enough. To protect the sign bit inside each quantized message, the traditional method is the static triple modular redundancy (TMR) scheme as applied in [13]. However, since two quantization bits are occupied for protecting the sign bit, the TMR scheme is not always beneficial for various storage error levels due to the loss of quantization precision. By analyzing the convergence process of LDPC decoding as well as referring to the results in [12, 14], it shows that when the magnitude of message is small, the precision bits, that is, the least significant bits (LSBs), are nonnegligible for decoding performance, while when the

message has a large magnitude value, the sign bit becomes even more critical for the residual errors.

Based on the aforementioned observations, we propose an adaptive embedded coding scheme for the unreliable messages to achieve a robust LDPC decoder. First, we put the messages into packages by taking advantage of the parallel message architecture of the quasi-cyclic (QC) LDPC decoders. The structure of message package permits more efficient block coding schemes for the sign bits other than simple TMR method. Then, two LSBs are adaptively employed for sign bits protection based on the magnitude level of message package. Moreover, we introduce a construction of Hamming product code for the adaptive coding, which has a multistage coding structure and outstanding error-correcting capability. We also discuss low complexity iterative decoding algorithms for the Hamming product code. Both theoretical analysis and Monte Carlo simulations demonstrate that the proposed adaptive message coding scheme outperforms the TMR scheme in decoding both thresholds and residual errors for various storage error levels.

The paper is organized as follows. In Section 2, the system models are introduced. Section 3 presents the discrete density evolution analysis on unreliable LDPC decoders. The adaptive message coding scheme and construction of Hamming product code are proposed in Section 4. We give the decoding algorithms for adaptive Hamming product codes in Section 5. Monte Carlo simulations are provided in Section 6. Section 7 concludes the entire paper.

2. System Models

2.1. LDPC Decoder. The hardware architecture of the QC-LDPC decoder is shown in Figure 1, which consists of interleaver ($\Pi_{H(\text{LDPC})}$), variable node units (VNU), check node units (CNU), and data buffers (RAM). Since the matrix of QC-LDPC is divided into subblocks, the decoders are always implemented with the partially parallel architecture [15–17], which means the messages of each subblock are calculated by the same VNU or CNU node in the pipeline operations. The constraint of the LDPC code is executed by the interleaver, which is used to deliver the messages between the VNU and CNU based on the parity-check matrix of LDPC in various decoding algorithms [18, 19]. To execute the BP decoding of LDPC, the decoder firstly obtains the log-likelihood (LLR) m_o from the channel. Then, VNU and CNU perform iterative computations, where the internal messages m_{v2c} and m_{c2v} are produced. Specifically, in VNU,

$$m_{v2c}(i) = m_o + \sum_{c' \in \mathcal{N}(v) \setminus c} m_{c'2v}(i-1), \quad (1)$$

while, in CNU,

$$m_{c2v}(i) = 2 \tanh^{-1} \left[\prod_{v' \in \mathcal{N}(c) \setminus v} \tanh \left(\frac{m_{v'2c}(i)}{2} \right) \right], \quad (2)$$

where $\mathcal{N}(v)$ and $\mathcal{N}(c)$ are defined as the sets of nodes connected to node v and node c , respectively. These messages are stored in memories during the decoding process. To

FIGURE 1: The partially parallel architecture of LDPC decoders.

implement LDPC decoders on integrated circuits, all of the messages will be quantized into bits. Existing studies [20] have shown that 4–6 bits' quantization on messages can provide ideal compromise between complexity and performance for LDPC decoders. Among the quantized bits, one is used for the sign, while the rest are used for the magnitude value.

2.2. Error Model of Memory. For the existing studies on LDPC decoders with faulty hardware, there are several widely accepted error models. As shown in Figure 2, where the model shown as Figure 2(a) is adopted in [4, 5, 7], the model shown as Figure 2(b) is adopted in [8, 9]. These models are both assumed to connect the error-free operation results with error channels, such as the binary symmetric channel (BSC) or the additive white Gaussian noise (AWGN) channel.

However, these two error models still have limitations for practical LDPC decoders. For example, the BSC error model is mostly utilized in bit flipping decoding algorithms, such as Gallager A decoding and Gallager B decoding, which make more sense in theoretical analysis. The AWGN error model is adopted in infinite-precision soft-decoding algorithms, where the messages are in continuous domain and assumed to be added with Gaussian noise by the faulty hardware.

In this paper, we consider the practical LDPC decoders, where finite-precision decoding algorithm is utilized. Following the studies in [11, 12], we assume a quantized BSC model for the storage errors, as shown in Figure 3. In the quantized BSC error model, the decoding messages are quantized into bits, each of which is assumed to pass a BSC error channel. The BSC errors for different bits are assumed to be independent, and the error ratios are assumed to be the same. The cross-over parameter α of the BSC channel is the flipping probability of the RAM cell, which is relevant to the radiation level and the service duration. As shown in Figure 1, there are 3 memories for the message storage: the LLR message storage, the V2C message storage, and the C2V message storage. In this paper, we assume the same bit flipping probability α_0 for all message memories.

FIGURE 2: Existing faulty storage models.

FIGURE 3: The quantized BSC model for the unreliable memories.

3. Analysis on LDPC Decoder with Unreliable Messages

3.1. Discrete Density Evolution. In this section, we define a discrete density evolution method for the analysis of finite-precision BP decoding of LDPC codes, which will give the performance of decoding thresholds and residual error ratios for LDPC decoders with different message protection schemes.

It has been proved by Varshney [4] that the symmetric conditions of density evolution are still suitable for the faulty LDPC decoders with symmetric hardware errors. Therefore, we can utilize the discrete density evolution by assuming all zero sequences are transmitted to analyze the finite-precision LDPC decoders with the BSC storage models. In this paper, only the regular (d_v, d_c) LDPC codes are considered for the sake of simplicity.

In the density evolution analysis, we define $\mathbf{P}_m^k = \{p_1, p_2, \dots, p_{2^Q-1}\}$ as the probability mass function (PMF) of the corresponding message m at iteration k , where Q is the number of quantization bits and p_i is the probability of the i th quantization symbol. For example, $\mathbf{P}_o$ is the PMF vector of the message m_o shown in Figure 1, which is the LLR from channel, while $\mathbf{P}_{v2c}^k$ is the PMF of message m_{v2c} at the k th decoding iteration.

Since the codewords are assumed to be all zero sequences, to initialize the discrete density evolution, $\mathbf{P}_o$ is calculated as

$$\mathbf{P}_o = \left\{ p_i = \frac{1}{\sqrt{2\pi} \cdot 2/\sigma} \int_{q_{i-1}}^{q_i} \exp\left(-\frac{(t + 2/\sigma^2)^2}{8/\sigma^2}\right) dt \right\}, \quad (3)$$

where q_i is the value of the i th quantization symbol, $q_0 = -\infty$, and $q_{2^Q-1} = +\infty$. Meanwhile, $\mathbf{P}_{c2v}^0$ is initialized as

$$\mathbf{P}_{c2v}^0 = \{0, \dots, 0, p_{2^{Q-1}} = 1, 0, \dots, 0\}. \quad (4)$$

After the initialization, the density evolution executes its iterations. Firstly, in the VNU nodes,

$$\mathbf{P}_v^k = \mathbf{P}_r^k \otimes (\mathbf{P}_{c2v}^{k-1})^{\otimes(d_v-1)}. \quad (5)$$

It is worth noting that, after the convolution operations, we shall combine the extra elements of $\mathbf{P}_v^k$ so as to ensure a length of $2^Q - 1$.

Secondly, in the CNU nodes, the magnitude values of messages are mapped into log-domain by function $\lambda(x) = -\log(\tanh(x/2))$. The corresponding PMF of the magnitude values is mapped by $\Lambda(\mathbf{P}_{v2c}^k)$. Further define

$$\mathbf{\Gamma}_{v2c}^k = [\text{sign}(\mathbf{P}_{v2c}^k), \Lambda(\mathbf{P}_{v2c}^k)]^{\otimes(d_c-1)}, \quad (6)$$

and thus the output PMF of CNU is updated by

$$\mathbf{P}_c^k = \text{sign}(\mathbf{\Gamma}_{v2c}^k) \cdot \Lambda(\text{abs}(\mathbf{\Gamma}_{v2c}^k)). \quad (7)$$

Similarly, the extra elements of $\mathbf{\Gamma}_{v2c}^k$ shall be combined after the convolution operations.

Finally, after the maximum iterations, the decoding decision is made in the VNU nodes, where the PMF is calculated as

$$\mathbf{P}_d^k = \mathbf{P}_r^k \otimes (\mathbf{P}_{c2v}^{k-1})^{\otimes d_v}, \quad (8)$$

and the probability of residual error p_e can be obtained by

$$p_e = \frac{1}{2} \cdot \mathbf{P}_d^k(2^{Q-1}) + \sum_{i=2^{Q-1}+1}^{2^Q-1} \mathbf{P}_d^k(i). \quad (9)$$

Above is the conventional discrete density evolution method for finite-precision LDPC decoders. However, this paper considers the issue of message storage errors, which means each message will suffer transformation of PMF outside the nodes. In the following, we will model the PMF transformation of unreliable message in density evolution.

Define $\mathbf{E} = \{e_1, e_2, \dots, e_Q\}$ as the quantization bit error vector, where e_i is the error probability of the i th quantization bit (e_1 for the sign bit, e_Q for the LSB). For example, we can make $e_1 = e_2 = \dots = e_Q = \alpha_0$ for the VRAM and CRAM

error models described in Section 2.2, where all quantized bits experience the same error probability. Further, define PMF transfer matrix $\Pi(\mathbf{E})$, which is calculated as follows:

$$\Pi(\mathbf{E}) = \left\{ p_{ij} = \prod_{k=1}^Q \beta_k(i, j) \right\}, \quad (10)$$

$$(i, j = 1, 2, \dots, 2^Q - 1),$$

where p_{ij} is the transfer probability from the i th quantization symbol s_i to the j th one s_j . And $\beta_k(i, j) = 1 - e_k$ if s_i and s_j have the same bit in the k th quantization position; otherwise $\beta_k(i, j) = e_k$. Since $\beta_k(i, j) = \beta_k(j, i)$, we know that $\Pi(\mathbf{E})$ is a symmetric matrix. As a result, the PMF transformations between RAM's input and output can be described as

$$\begin{aligned} \mathbf{P}_r &= \mathbf{P}_o \cdot \Pi(\mathbf{E}(\alpha_0)), \\ \mathbf{P}_{v2c}^k &= \mathbf{P}_v^k \cdot \Pi(\mathbf{E}(\alpha_0)), \\ \mathbf{P}_{c2v}^k &= \mathbf{P}_c^k \cdot \Pi(\mathbf{E}(\alpha_0)). \end{aligned} \quad (11)$$

We could set various error vectors $\mathbf{E}$ for corresponding protection schemes for unreliable messages in discrete density evolution, which will give asymptotic performance of different protection schemes.

3.2. Analysis on Various Bit Errors for Finite-Precision Messages. Based on the discrete density evolution method defined in Section 3.1, a threshold analysis is provided to demonstrate the various effects of finite-precision message bits. It is shown that the sign bits *have the most influence* on the decoding thresholds of LDPC codes.

We execute the discrete density evolution on a (4, 32) regular LDPC code with the 6 bits' quantized decoder in this paper. To analyze the various effects of quantized bits, we set $\mathbf{E} = \{0, \alpha_0, \alpha_0, \alpha_0, \alpha_0, \alpha_0\}$ for the one highest bit protected memory error model, which means the sign bit is assumed to be error-free. Similarly, several k highest bits protected models can be defined, where $\mathbf{E} = \{0, \dots, 0, \alpha_0, \dots, \alpha_0\}$. The decoding thresholds are obtained by the discrete density evolution, as shown in Figure 4. We can see that if the sign bit is protected, the threshold will not be severely affected, while additional protection on the extra bits can provide little gain.

3.3. Triple Modular Redundancy Scheme for Sign Bit Protection. For LDPC decoders, the cost is overwhelming to protect every message bit. However, as mentioned before, it is not necessary since the sign bits are demonstrated to be the most important. Thus, following the idea of unequal error protection [21], we can simply set protection on the sign bits to promise a low complexity. In this section, we will firstly introduce traditional TMR protection scheme for the sign bits and then discuss its advantages and disadvantages.

As in [13], TMR has been applied in protecting the messages for LDPC decoder on unreliable hardware. However, TMR will charge two extra bits for protecting the sign bit while the messages are typically quantized into only 4 to 6 bits [20]. As a result, if we maintain the quantity of

FIGURE 4: The thresholds of protecting various message bits.

message quantization bits under the constraint of complexity, introducing TMR will bring a loss of quantization precision, which is not always beneficial for various storage error ratios. In the following, using the discrete density evolution method described in Section 3.1, we will analyze the performance of TMR scheme for the sign bit protection. We set $Q = 6$ for the quantity of quantization bits, which is adopted for most practical LDPC decoders. Moreover, to model the storage error of the TMR-protected messages, the error vector is set to be $\mathbf{E}_t = \{3\alpha_0^2 - 2\alpha_0^3, \alpha_0, \alpha_0, \alpha_0\}$, which is quantized with 4 bits actually. While the unprotected LDPC decoder is set to be $\mathbf{E}_u = \{\alpha_0, \alpha_0, \alpha_0, \alpha_0, \alpha_0, \alpha_0\}$, the results of discrete density evolution under different storage error ratio α_0 are shown in Figure 5.

From the analysis, it can be observed that when the storage error ratio is high (e.g., $\alpha_0 = 10^{-3}$), the LDPC decoder without protection cannot work anymore, while the TMR-protected one can work with a dramatic degradation of decoding threshold. However, when the storage error ratio is low enough, for $\alpha_0 = 10^{-4}$ and $\alpha_0 = 10^{-5}$, due to the loss of quantization precision, the TMR-protected LDPC decoders will show disadvantages in decoding threshold compared with the unprotected ones. However, the TMR protection scheme still has its advantages: we can observe that TMR-protected decoders have lower decoding residual errors for all levels of storage error ratios.

3.4. Existing Adaptive Messages Coding Scheme. We noticed that a similar adaptive coding scheme for approximate computing with faulty storage has been proposed in [22]. In [22], an adaptive message coding scheme on faulty min-sum LDPC decoders is mentioned. In detail, when the messages were written into RAMs, if the MSB was 1, the last two LSBs were neglected, while the corresponding memory addresses were used for a (3, 1) repetition coding on the sign bit. Otherwise, the messages would be stored in the RAMs directly. When the LDPC decoder reads a message from the RAMs, the MSB

FIGURE 5: Performance analysis under different storage error ratios.

was checked. If the MSB was read as 1, a decoding on the (3, 1) code was executed to obtain the sign bit, while the last two LSBs were selected from $\{0, 1\}$ randomly. Otherwise, the messages were assigned with the read values.

The aforementioned scheme makes full use of the LSBs in the messages. It has efficiently protected the unreliable messages without using any storage redundancy. However, there are some disadvantages for this protection scheme. Firstly, the adaptive coding is executed inside the single message, which is typically quantized with no more than 7 bits for the reason of complexity [20]. Consequently, there are not enough bits for the efficient coding schemes. For example,

when the number of quantization bits is from 4 to 6, only simple repetition codes can be utilized. And this scheme is even inapplicable when the messages are quantized into less than 4 bits. Secondly, whether the adaptive coding is executed or not is totally based on the MSB, which is also subject to the storage errors. In such a case, the decoding of the adaptive code may be incorrectly executed, which further degrades the performance of the sign bit protection.

We demonstrate the exact error-correcting performance of the sign bits for this coding scheme as follows. In the first case where the MSB is 1, the encoding will be executed. If the MSB is read correctly, the (3, 1) code will be properly decoded

FIGURE 6: Structure of adaptive package coding.

with an output error ratio of $3\alpha^2 - 2\alpha^3$. If the MSB is read in error, the decoding will be neglected, which results in an error rate of α for the sign bit. That means that the expectation of the error rate for the sign bit is

$$\begin{aligned} p_e(\text{MSB} = 1) &= (1 - \alpha) \times (3\alpha^2 - 2\alpha^3) + \alpha \times \alpha \\ &= 4\alpha^2 - 5\alpha^3 + 2\alpha^4. \end{aligned} \quad (12)$$

In the second case where the MSB is 0, similarly the error rate should be calculated with two cases, which is derived by

$$\begin{aligned} p_e(\text{MSB} = 0) &= (1 - \alpha) \times \alpha + \alpha \times \left[\frac{1}{4}(1 - \alpha) + \frac{3}{4}\alpha \right] \\ &= \frac{5}{4}\alpha - \frac{1}{2}\alpha^2. \end{aligned} \quad (13)$$

Unfortunately, since the storage error probability α is small enough, when the MSB is 1, this coding scheme cannot achieve the error-correcting capability of the (3, 1) repetition code, while when the MSB is 0, the error probability of the sign bit is even higher than the one without protection.

4. Adaptive Message Coding Scheme

In this section, we firstly present the architecture of the proposed adaptive coding scheme. Then, a specific construction of Hamming product code for the adaptive strategy is provided. Next, we analyze the performance of the proposed scheme theoretically.

4.1. Protecting Sign Bits Utilizing LSBs Adaptively. As analyzed in Section 3.4, protecting the sign bits of unreliable messages by occupying extra bits is not always the best scheme. The degradation of decoding threshold is mainly caused

by the loss of quantization precision. However, we notice that quantization precision affects decoding performance specifically when the magnitude of message is small; that is, when the message has a large magnitude, the LSBs are less important. On the other hand, with the convergence of LDPC decoding process, the sign bits of messages are shown to have significant effects on the decoding residual errors when most messages have large magnitudes. Based on these observations, while referencing the idea of adaptation as in [23], we introduce an adaptive scheme for protecting the sign bits of unreliable messages. The basic idea is that when the message magnitude is small, the LSBs are used for maintaining quantization precision, while when the magnitude is large enough, the storage space for LSBs is used to protect sign bits to ensure a lower residual error.

What is more, existing studies execute protection on each single message, where only simple coding scheme (such as TMR) can be utilized. However, we notice that LDPC decoders are usually implemented with a partially parallel architecture, as described in Section 2.1. In other words, a group of messages are produced simultaneously. It inspires us to put the sign bits into packages so that we can introduce efficient block coding schemes instead of the traditional TMR.

As shown in Figure 6, the structure of our proposed adaptive coding scheme is described as follows: first, put k concurrently produced messages into a package. Then, define T_1 and T_2 as the adaptive thresholds, where $0 < T_1 < T_2 < L_{\max}$ ($L_{\max}$ is the maximum absolute value of quantization). Next, when the messages are written into RAMs, for each message package, calculate the average magnitude value t of the k messages. Based on the value of t , the adaptive coding is divided into 3 stages as below.

FIGURE 7: The (48, 16) Hamming product code.

- (i) If $0 \leq t < T_1$, all LSBs of the message package are reserved for quantizing messages.
- (ii) If $T_1 \leq t < T_2$, the storage space for the last k LSBs are occupied for coding on the k sign bits with a code rate of $1/2$.
- (iii) If $T_2 \leq t \leq L_{\max}$, the storage space for the last $2k$ LSBs are occupied for coding on the k sign bits with a code rate of $1/3$.

Reversibly, when the messages are read from RAMs, adaptive decoding is executed based on the value of t . If the storage of LSBs has been occupied for the sign bits, the LSBs of messages are randomly assigned.

4.2. Construction of Adaptive Hamming Product Code. In this section, we will give a specific code construction for the adaptive coding scheme.

To adaptively protect the sign bits of message packages, the ideal block code should have the features of multistage coding structure, as well as low coding complexity and appropriate block length. We introduce Hamming product codes as the adaptive package codes based on the following advantages. First, the product codes are constructed by several subcodes, whose coding process can be easily designed into multistage. Second, Hamming codes have the simplest decoders and encoders among all of the block codes, which only consist of several basic logic gates. Moreover, as the data is usually operated in bytes, where each byte contains 8 bits, in order to make the package codes suitable for the data operations, we choose the modified Hamming product code, which is $(n, k) = (48, 16)$. It is worth noting that some other short algebraic codes could be adopted to constitute the product code, such as Gray codes in [24], at the cost of complexity.

As shown in Figure 7, the dark points are the sign bits in one message package and the white points are the LSBs. The row and column subcodes are both (8, 4) Hamming codes. For such multistage (48, 16) Hamming product codes, package size $k = 16$, the first coding stage is that both row

and column subcodes are inactive when $0 \leq t < T_1$, while the second coding stage is executed by only activating the row subcodes when $T_1 \leq t < T_2$. And the third coding stage is executed by activating the whole subcodes when $T_2 \leq t \leq L_{\max}$.

4.3. Theoretical Analysis. In this section, we will also utilize the discrete density evolution method to analyze our proposed adaptive package coding scheme. As mentioned before, we should deduce the error vector $\mathbf{E}$ for the proposed scheme.

As defined in Section 4.1, since the stage of adaptive coding is based on the average magnitude values of message packages, we should firstly calculate the PMF of the summation of magnitude values in one message package. First, the PMF of message magnitudes $|\mathbf{P}|$ is derived as

$$|\mathbf{P}| = \left\{ \mathbf{P} \left(2^{(Q-1)} \right), \mathbf{P} \left(2^{(Q-1)} - 1 \right) + \mathbf{P} \left(2^{(Q-1)} + 1 \right), \dots \right\}. \quad (14)$$

Then, the PMF of the summation of k magnitudes can be obtained by

$$\mathbf{P}_{\text{sum}} = (|\mathbf{P}|)^{\otimes k}. \quad (15)$$

Based on the PMF of magnitude summation, we can obtain the probabilities of the 3 stages of adaptive coding, respectively, by

$$\begin{aligned} p_{t0} &= \sum_{\text{mag}(i)=0}^{k \cdot T_1} \mathbf{P}_{\text{sum}}(i), \\ p_{t1} &= \sum_{\text{mag}(i)=k \cdot T_1}^{k \cdot T_2} \mathbf{P}_{\text{sum}}(i), \\ p_{t2} &= \sum_{\text{mag}(i)=k \cdot T_2}^{k \cdot L_{\max}} \mathbf{P}_{\text{sum}}(i), \end{aligned} \quad (16)$$

where $\text{mag}(i)$ is the corresponding magnitude value of the i th element of $\mathbf{P}_{\text{sum}}$. Next, we need to calculate the error ratio for each stage of the adaptive Hamming product codes. As for (n, k, d) block codes with a raw error ratio of α_0 , the error upper bound is derived by

$$e_{\text{blk}}(n, k, d, \alpha_0) = \sum_{i=\lfloor (d+1)/2 \rfloor}^n \binom{n}{i} (\alpha_0)^i (1 - \alpha_0)^{n-i}. \quad (17)$$

Based on (17), the error vector for the first coding stage is $\mathbf{E}_{t0} = (\alpha_0, \alpha_0, \alpha_0, \alpha_0, \alpha_0, \alpha_0)$, while for the second stage it is $\mathbf{E}_{t1} = (e_{\text{blk}}(8, 4, 4, \alpha_0), \alpha_0, \alpha_0, \alpha_0, \alpha_0, 0.5)$, and for the third stage it is $\mathbf{E}_{t2} = (e_{\text{blk}}(48, 16, 7, \alpha_0), \alpha_0, \alpha_0, \alpha_0, 0.5, 0.5)$. As a result, the eventual error vector is obtained by

$$\mathbf{E} = p_{t0} \cdot \mathbf{E}_{t0} + p_{t1} \cdot \mathbf{E}_{t1} + p_{t2} \cdot \mathbf{E}_{t2}. \quad (18)$$

We set $T_1 = 0.4L_{\max}$ and $T_2 = 0.8L_{\max}$ and analyze the performance of our proposed scheme with $\alpha_0 = 10^{-3}$, $\alpha_0 = 10^{-4}$, and $\alpha_0 = 10^{-5}$. Firstly, we simulate with the parameter

FIGURE 8: Performance analysis on adaptive package coding scheme with $(d_v, d_c) = (4, 32)$.

$(d_v, d_c) = (4, 32)$, whose results are shown in Figure 8. Then, as a comparison, we set $(d_v, d_c) = (4, 8)$ to verify the effects of different row weights (or the coding rates) on the performance, the results of which are shown in Figure 9. We can see that, with the proposed adaptive package coding scheme, the performances of both decoding threshold and residual error are significantly improved. What is more, our proposed scheme is effective in different coding rates.

5. Decoding of Hamming Product Codes

The Hamming product code we have introduced has an outstanding minimum distance characteristic. However, its error-correcting capability can only be achieved under the maximum likelihood (ML) decoding, which has high complexity and is not practical for LDPC message protection. In this section, we will discuss specific decoding algorithms

for the Hamming product code, which achieves good performance with low complexity.

5.1. Iterative Decoding of the Hamming Product Code. For the subcode $(8, 4)$ Hamming code, the Hamming distance is 4; that is, it can correct any one-bit error in one block. But when there are two error bits, the decoder can only declare a block error without locating the error bits. Based on the Hamming codes' characters of both error correcting and detecting, we define two states for the output bits of the Hamming product decoder: the fixed bits and the erasure bits. The decoding algorithm of the Hamming product code is described as follows:

- (i) Iterative step: the row subcodes and the column subcodes execute their decoding algorithms iteratively. During the decoding, if the Hamming decoder cannot locate the error bits, keep the block unchanged;

FIGURE 9: Performance analysis on adaptive package coding scheme with $(d_v, d_c) = (4, 8)$.

otherwise, update the block. After several iterations (we set it to 2 iterations here), stop the iterative decoding.

- (ii) Decision step: firstly, the error detecting is executed by the Hamming decoders. Define $\mathcal{R} \subset \{1, 2, 3, 4\}$ as the set of indices where the row subcodes detect block errors. Similarly, define index set $\mathcal{C}$ for the column subcodes. Then, declare the bits located at (i, j) in the 4×4 information bit matrix as erasure bits, where $i \in \mathcal{R}$ and $j \in \mathcal{C}$. The rest of the bits are declared as fixed bits.

We utilize a low-order approximation method to evaluate the performance of the Hamming product code with the proposed decoding algorithm. Since two states are defined

for the output bits, we use two parameters to describe the decoding performance: the bit erasure ratio r_e and the bit error ratio r_p . Both parameters can be approximately derived by the most likely error patterns. The probability of the i th-order error pattern can be derived as

$$P(i) = \alpha_0^i (1 - \alpha_0)^{48-i}, \quad (19)$$

where α_0 is the bit flipping probability of the RAM. As listed in Table 1, the error patterns with orders lower than 4 are analyzed. Apparently, if there are less than two-bit errors in a block of the Hamming product code, no erasure or error bit exists. Therefore, only the error patterns with 3rd order and 4th order are adopted to deduce the approximate

TABLE 1: Low-order error pattern analysis.

Error pattern order	Bit erasures total	Bit errors total
1	0	0
2	0	0
3	256	16
4	13008	1680

performance. Using the 3rd order only, the error performance can be derived by

$$\begin{aligned}\tilde{r}_e(3) &= \frac{256}{16} \times P(3) = 16\alpha^3(1-\alpha)^{45}, \\ \tilde{r}_p(3) &= \frac{16}{16} \times P(3) = \alpha^3(1-\alpha)^{45}.\end{aligned}\quad (20)$$

While using both the 3rd order and 4th order, the performance is derived as follows:

$$\begin{aligned}\tilde{r}_e(3,4) &= \frac{256}{16} \times P(3) + \frac{13008}{16} \times P(4) \\ &= 16\alpha^3(1-\alpha)^{45} + 813\alpha^4(1-\alpha)^{44}, \\ \tilde{r}_p(3,4) &= \frac{16}{16} \times P(3) + \frac{1680}{16} \times P(4) \\ &= \alpha^3(1-\alpha)^{45} + 105\alpha^4(1-\alpha)^{44}.\end{aligned}\quad (21)$$

Meanwhile, Monte Carlo simulations for the iterative decoding of Hamming product code are provided. The curves of the performance are shown in Figure 10. We can see that $\tilde{r}_e(3,4)$ and $\tilde{r}_p(3,4)$ are very close to the results of Monte Carlo simulations. Moreover, the Hamming product code outperforms the traditional TMR scheme dramatically.

5.2. Enhanced Decoding of the Hamming Product Code. In fact, the performance of Hamming product code can be further improved at the expense of complexity for the iterative decoding. In this section, an enhanced decoding scheme is proposed to obtain better performance by introducing more decision logics.

As analyzed in Section 5.1, the 3rd-order error pattern has the most influence on the decoding performance of the Hamming product code. Specifically, the most typical error patterns for decoding erasure and error are depicted in Figure 11. Without loss of generality, we can assume that the row subcodes are decoded firstly in the iterative step. If there are three check bit errors (denoted by the dark points in Figure 11(a)) in one column subcode, one of the column subcode's information bits (denoted by the point marked with an asterisk) will be incorrectly decoded. In such a case, when it comes to the decision step, this incorrect information bit, together with the other three incorrect check bits, will constitute a valid codeword for the column subcode. Thus, it will result in a decoding error. Similarly, as shown in Figure 11(b), if there are three errors (denoted by the dark points) located, respectively, in the row subcode and the column subcode, they will simultaneously disable

FIGURE 10: The low-order approximation of r_e and r_p .

the decoding of the row and column subcodes. In such a case, the error in the intersection position will be declared as an erasure bit according to the decision logic of the aforementioned decoding algorithm.

As a matter of fact, the decoding erasure bits and error bits caused by the 3rd-order error patterns all occurred in the similar ways mentioned above. Based on this analysis, an enhanced decoding scheme is proposed by adding the following two decision logics in the decision step:

- (i) 3rd-order error bit decision: after the error detecting, if the index sets $\mathcal{R} = \{i\}$ and $\mathcal{C} = \{j\}$ are both single element sets, the bit located at (i, j) is flipped and declared as a fixed bit.
- (ii) 3rd-order erasure bit decision: if one bit is decoded into different values by the row subcode and the column subcode, it is declared as an erasure bit.

The performance of the enhanced decoding scheme is shown in Figure 12. There is an improvement for both r_e and r_p . It should be noted that the additional decision logics are only used to cope with the 3rd-order error patterns. In fact, more decision logics for the higher-order error patterns can be introduced to obtain better performance.

5.3. Decoding Complexity of the Hamming Product Code. Another key issue is the complexity of decoding Hamming product code compared with traditional TMR scheme. As TMR only consumes a majority decision logic module to decode the duplicate check, it is generally believed that introducing advanced long block codes will definitely increase the hardware complexity. However, in this section, based on the Field Programmable Gate Array (FPGA) implementation, we will see that the hardware complexity of Hamming product code can be even lower than TMR scheme in some cases. Moreover, there will be a flexible tradeoff between hardware

FIGURE 11: The 3rd-order error patterns.

FIGURE 12: The performance of the enhanced decoding scheme.

consumption and decoding delay for the Hamming product code.

In applications, LDPC encoder and decoder are mostly implemented based on FPGA, which is reconfigurable and widely adopted in communication systems. A major difference between FPGA and the Application Specific Integrated Circuit (ASIC) is the structure of combinational logic circuit. For FPGA, the combinational logic is not composed of actual logic gates. Instead, it is based on a structure called Lookup Table (LUT), which is actually a small block of RAM. The input of combinational logic is connected to the RAM's address, and the logical output is presynthesized and stored into the RAM. Thus arbitrary logical operation can be implemented by looking up into the storage for each input logic combination. For conventional FPGA, 4-input

and 6-input LUTs are mostly equipped. As a result, the TMR decision is actually processed by a 4-input LUT on FPGA. Next, we will compare the consumption of LUTs for the TMR and proposed schemes. In our proposed adaptive message coding scheme, each 16 messages are grouped into one package. Consequently, the corresponding consumption for TMR scheme is 16 4-input LUTs totally. Comparatively, the consumption of the proposed scheme is shown in Figure 13. We can see that, for the (8, 4) Hamming encoder, only four 4-input LUTs are required, while, for the decoder, four 4-input LUTs are utilized to generate the correctors, and then each decoded information bit outputs through a 6-input LUT by logically processing the correctors and original value. To sum up, the total consumption is eight 4-input and four 6-input LUTs for (8, 4) Hamming encoder and decoder. Based on these analyses, the hardware complexity of proposed Hamming product code is no more than TMR scheme, while roughly speaking it even consumes less resources on FPGA. Actually, in our iterative decoding algorithm for the Hamming product codes, the cost for improving error-correcting performance of unreliable message is decoding delay instead of hardware complexity. As the subcodes of Hamming product code are decoded iteratively, the decoding of each message package will occupy a certain number of clocks. Thus if the LDPC decoder is poor in timing margin, the iterative decoding of Hamming product code will severely degrade the decoding throughput. Fortunately, the subcodes of Hamming product code can be decoded in parallel, which means we can compress the decoding clocks by parallel processing with multiple Hamming decoders. In this case, there can be a flexible tradeoff between hardware complexity and decoding delay. The specific consumption of space-time resource for various arrangements is shown in Table 2.

6. Simulations

In this section, Monte Carlo simulations are executed on the finite length codewords of LDPC. We utilize the (8176, 7154) LDPC code defined by CCSDS in [25], which is publicly

FIGURE 13: The implementation structure of various schemes based on LUT.

TABLE 2: Tradeoff between complexity and decoding delay.

Serial number	Consumption of LUTs	Decoding clocks
1	4 inputs $\times$ 4 + 6 inputs $\times$ 4	8
2	4 inputs $\times$ 8 + 6 inputs $\times$ 8	4
3	4 inputs $\times$ 16 + 6 inputs $\times$ 16	2

available and has outstanding performance. In the simulations, the messages are quantized into 6 bits, while the maximum number of iterations is set to 15. The communication channel is assumed to be the additive white Gaussian noise (AWGN) channel. To demonstrate the effectiveness of our proposed scheme under various storage error levels, the flipping probability of BSC model is set from $\alpha_0 = 10^{-3}$ to $\alpha_0 = 10^{-4}$. We compare the adaptive message coding scheme (labeled as “proposed”) with both the traditional TMR scheme (labeled as “TMR”) and the one without protection (labeled as “no sch”). The results are shown in Figure 14. We can see that when $\alpha_0 = 10^{-3}$, the proposed scheme has a gain of 0.2 dB compared to TMR scheme, while the unprotected one even cannot work. When $\alpha_0 = 10^{-4}$, The proposed scheme still outperforms the other schemes.

7. Conclusion

This paper considered the challenge of implementing LDPC decoders on unreliable memories. We explored the effects of various message bits on finite-precision LDPC decoders and introduced an effective adaptive coding scheme based on the magnitude level of messages. We put the messages into packages and proposed a Hamming product code to adaptively correct the sign bits, as well as discussing two low complexity decoding algorithms. The discrete density evolution analysis showed that the proposed scheme outperforms traditional TMR scheme in decoding both threshold and

FIGURE 14: Simulations on the (8176, 7154) LDPC code with message storage errors.

residual errors under various storage error levels. Moreover, Monte Carlo simulations showed that the proposed scheme could at least obtain a gain of 0.3 dB to the static TMR scheme when the storage error probability was from 10^{-3} to 10^{-4} .

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC 91538203), the new strategic industries development projects of Shenzhen City (JCYJ20150403155812833), and the Beijing Innovation Center for Future Chips, Tsinghua University.

References

- [1] K. S. Andrews, D. Divsalar, S. Dolinar, J. Hamkins, C. R. Jones, and F. Pollara, "The development of turbo and LDPC codes for deep-space applications," *Proceedings of the IEEE*, vol. 95, no. 11, pp. 2142–2156, 2007.
- [2] Q. Huang, Q. Xiao, L. Quan, Z. Wang, and S. Wang, "Trimming soft-input soft-output viterbi algorithms," *IEEE Transactions on Communications*, vol. 64, no. 7, pp. 2952–2960, 2016.
- [3] D. Stone, A. Lindenmoyer, G. French et al., "NASA's approach to commercial cargo and crew transportation," *Acta Astronautica*, vol. 63, no. 1-4, pp. 192–197, 2008.
- [4] L. R. Varshney, "Performance of LDPC codes under faulty iterative decoding," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4427–4444, 2011.
- [5] F. Leduc-Primeau and W. J. Gross, "Faulty Gallager-B decoding with optimal message repetition," in *Proceedings of the 2012 50th Annual Allerton Conference on Communication, Control, and Computing*, Allerton 2012, pp. 549–556, USA, October 2012.
- [6] C. H. Huang, Y. Li, and L. Dolecek, "Gallager B LDPC decoder with transient and permanent errors," *IEEE Transactions on Communications*, vol. 62, no. 1, pp. 15–28, 2014.
- [7] S. M. S. Tabatabaei Yazdi, H. Cho, and L. Dolecek, "Gallager B decoder on noisy hardware," *IEEE Transactions on Communications*, vol. 61, no. 5, pp. 1660–1673, 2013.
- [8] C. H. Huang, Y. Li, and L. Dolecek, "Noisy belief propagation decoder," in *Proceedings of the 48th Asilomar Conference on Signals, Systems and Computers*, ACSSC 2015, pp. 2111–2115, USA, November 2014.
- [9] C. H. Huang, Y. Li, and L. Dolecek, "Belief propagation algorithms on noisy hardware," *IEEE Transactions on Communications*, vol. 63, no. 1, pp. 11–24, 2015.
- [10] E. Dupraz, D. Declercq, B. Vasic, and V. Savin, "Analysis and Design of Finite Alphabet Iterative Decoders Robust to Faulty Hardware," *IEEE Transactions on Communications*, vol. 63, no. 8, pp. 2797–2809, 2015.
- [11] C. K. Ngassa, V. Savin, and D. Declercq, "Min-Sum-based decoders running on noisy hardware," in *Proceedings of the 2013 IEEE Global Communications Conference, GLOBECOM 2013*, pp. 1879–1884, USA, December 2013.
- [12] A. Balatsoukas-Stimming and A. Burg, "Density evolution for min-sum decoding of LDPC codes under unreliable message storage," *IEEE Communications Letters*, vol. 18, no. 5, pp. 849–852, 2014.
- [13] M. May, M. Alles, and N. Wehn, "A case study in reliability-aware design: A resilient LDPC code decoder," in *Proceedings of the Design, Automation and Test in Europe, DATE'08*, pp. 456–461, 2008.
- [14] C. H. Huang, Y. Li, and L. Dolecek, "Adaptive error correction coding scheme for computations in the noisy min-sum decoder," in *Proceedings of the IEEE International Symposium on Information Theory, ISIT'15*, pp. 1906–1910, June 2015.
- [15] Q. Li, X. Qu, L. Yin, and J. Lu, "Generalized Low-Density Parity-Check coding scheme with Partial-Band Jamming," *Tsinghua Science and Technology*, vol. 19, no. 2, pp. 203–210, 2014.
- [16] Z. Chen, L. Yin, Y. Pei, and J. Lu, "CodeHop: physical layer error correction and encryption with LDPC-based code hopping," *Science China Information Sciences*, vol. 59, no. 10, Article ID 102309, 2016.
- [17] P. Wang, L. Yin, and J. Lu, "An efficient helicopter-satellite communication scheme based on check-hybrid ldpc coding," *Tsinghua Science and Technology*, pp. 10–26599, 2018.
- [18] Q. Huang, L. Song, and Z. Wang, "Set Message-Passing Decoding Algorithms for Regular Non-Binary LDPC Codes," *IEEE Transactions on Communications*, 2017.
- [19] Q. Huang, M. Zhang, Z. Wang, and L. Wang, "Bit-reliability based low-complexity decoding algorithms for non-binary LDPC codes," *IEEE Transactions on Communications*, vol. 62, no. 12, pp. 4230–4240, 2014.
- [20] J. Lee and J. Thorpe, "Memory-efficient decoding of LDPC codes," in *Proceedings of the Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, pp. 459–463, Adelaide, Australia, September 2005.
- [21] J. Huang, Z. Fei, C. Cao, M. Xiao, and D. Jia, "On-Line Fountain Codes with Unequal Error Protection," *IEEE Communications Letters*, vol. 21, no. 6, pp. 1225–1228, 2017.
- [22] C. H. Huang, Y. Li, and L. Dolecek, "ACOCO: Adaptive Coding for Approximate Computing on Faulty Memories," *IEEE Transactions on Communications*, vol. 63, no. 12, pp. 4615–4628, 2015.
- [23] X. Ji, J. Xu, Y. L. Che, Z. Fei, and R. Zhang, "Adaptive Mode Switching for Cognitive Wireless Powered Communication Systems," *IEEE Wireless Communications Letters*, vol. 6, no. 3, pp. 386–389, 2017.
- [24] L. Wang, Z. Wang, Q. Huang, and M. Zhang, "Balanced Gray Codes with Flexible Lengths," *IEEE Communications Letters*, vol. 20, no. 5, pp. 894–897, 2016.
- [25] CCSDS, "Low density parity check codes for use in near-earth and deep space applications," CCSDS, 2011.