# Evaluation of Error-Correcting Codes for Radiation-Tolerant Memory

**Seungjune Jeon,* B. V. K. Vijaya Kumar,* Euiseok Hwang,* and Michael K. Cheng†**

In space, radiation particles can introduce temporary or permanent errors in memory systems. To protect against potential memory faults, either thick shielding or error-correcting codes (ECC) are used by memory modules. Thick shielding translates into increased mass, and conventional ECCs designed for memories are typically capable of correcting only a single error and detecting a double error. Decoding is usually performed through hard decisions where bits are treated as either correct or flipped in polarity. We demonstrate that low-density parity-check (LDPC) codes that are already prevalent in many communication applications can also be used to protect memories in space. Because the achievable code rate monotonically decreases with time due to the accumulation of permanent errors, the achievable rate serves as a useful metric in designing an appropriate ECC. We describe how to compute soft symbol reliabilities on our channel and compare the performance of soft-decision decoding LDPC codes against conventional hard-decision decoding of Reed-Solomon (RS) codes and Bose-Chaudhuri-Hocquenghem (BCH) codes for a specific memory structure.

## I. Introduction

Errors in memories can be soft (i.e., transient) or hard (i.e., permanent) errors. Soft errors may be caused by energetic particles, coupling from power supply noise, or variability in device behavior. Typically, soft errors would produce only a single-cell malfunction. As memory building blocks shrink into the nanometer regime and when memories are used in space applications, the frequency of multiple-cell malfunction increases, and these error events can range from a few bit flips to hundreds of errors. In addition to soft errors, hard errors can occur both in manufacturing due to defects or lithography contaminants, and in use due to device wear-out or cosmic radiation.

For space applications, memories are either made radiation-hard by strong material shielding or made radiation-tolerant through protection of error-correcting codes (ECCs). However, current radiation-tolerant approaches may not handle multiple bit errors elegantly or at all. For example, the Mars Exploration Rovers (MERs) adopted a single-error-correction

* Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania.

† Communication Architectures and Research Section.

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. 

and double-error-detection code to protect memory accesses. This simple error detection and correction (EDAC) code cannot sustain an event with three or more bit errors. Moreover, the access period to selected memory modules on the MERs was about 90 ns, or on the order of 11 MHz, because the EDAC circuit was implemented on a radiation-hard chip and clock rates on these elements are limited. In this article, we consider modern low-density parity-check (LDPC) codes for memory systems. LDPC codes are capacity-approaching codes that are increasingly being adopted in communications systems ranging from wireless routers to space communications. However, LDPC codes have yet to be considered for protecting memories because well-performing short (less than 1 kbit information length) LDPC codes are hard to design and the complexity of iteratively decoding long LDPC codes is higher than the decoding of conventional codes. Decoding LDPC codes also uses soft symbol information provided by the channel, but memory outputs are generally treated as hard bits and not soft information. In order to obtain the full performance of soft-decision decoding LDPC codes, we introduce a technique to generate soft symbol information using parameters that indicate the strength of radiation and the period of rewrites of the memory contents. We show that LDPC codes can improve, when compared to conventional Reed-Solomon (RS) codes, the reliability of commercial off-the-shelf (COTS) memory systems targeted for space use.

ECCs for memories in space have been studied in literature. Goodman et al. [1] evaluated single-error-correction (SEC) codes. Saleh et al. [2] investigated single-error-correction and double-error-detection (SECDED) codes. Shirvani et al. [3] proposed using RS codes to protect against data corruption in software. Cardarilli et al. [4] proposed an analytical method to calculate block error rates of RS codes for scrubbing memory systems. The same authors [5] also suggested adaptively increasing the length of RS codes to track changing radiation environments. Kaneko et al. [6] made measurements to characterize memory failure events in a radiation environment. Recently, MacLeod et al. [7] discussed a plan for a satellite test of a nonvolatile memory device. Nguyen and Irom [8] tested radiation effects on recently developed COTS NAND flash memory devices for both single-level cell and multilevel cell devices. The authors [9] demonstrated that 1-kbit LDPC codes can outperform RS codes in the memory systems in a radiation environment.

In this article, we extend the work described above. In Section II, we define a new channel to model both soft and hard errors in space and discuss the channel capacity of such channels briefly. In Section III, we apply RS, Bose-Chaudhuri-Hocquenghem (BCH), and LDPC codes to our memory model. For decoding LDPC codes, we derive the needed soft information from our channel model. In Section IV, we compare RS, BCH, and LDPC code performance in the targeted environment. In Section V, we summarize our work.

## II. Model of Memory Systems

### A. Channel Model

There are two types of errors in memory systems affected by radiation. A *soft error* is temporary such that the memory content at the soft error location has been changed from the original but can be corrected. For example, ECCs can correct soft errors by updating the erroneous location with the correct value. Therefore, the number of soft errors can be decreased after the memory contents are overwritten by the ECC decoding output.

In contrast, a *hard error* is fixed such that the memory content at the hard error location cannot be changed. No new information can be written. The locations of hard errors can be detected and made known to the ECC decoder. Therefore, hard errors can be treated as erasures. The number of hard errors cannot be decreased even after the memory contents are overwritten by the ECC decoding output, and hard errors will only increase with time.

*Scrubbing* is an operation to refresh memory content with the ECC decoder output to correct errors. The *scrubbing interval* is the time between each scrubbing. A scrubbing interval is denoted by $T_s$ in this article. The number of soft errors can be reduced after scrubbing whereas the number of hard errors cannot be decreased by scrubbing. The number of hard errors only accumulates. Frequent scrubbing or short scrubbing interval may keep error rates low over longer periods. However, frequent scrubbing leads to high hardware power consumption, which is unattractive in spacecraft.

A way to circumvent hard error locations is to write the corrected bits in undamaged locations after scrubbing. However, this scheme requires additional memories to store both the corrected bits and their locations. Moreover, the new bit locations and the data of addresses are also not immune to radiation effects. Since this scheme uses more redundant bits, the effectiveness of this scheme should be evaluated by comparing ECCs of lower code rates. Analysis and evaluation of this scheme is out of the scope of this article. We will focus on nominal scrubbing without relocating hard error bits.

Since the number of errors caused by radiation particles depends on the time interval $T$ of the radiation exposure, our channel model depends on $T$, as seen in Figure 1(a). COTS memories are often used in spacecraft, but COTS memories do not provide soft information per bit location during readback. The available information to the ECC decoder is the binary information and the location of hard errors. We mark erasures as $\epsilon$ and define the transition probabilities as follows:

$$\Pr\big(Y = 1 \,\big|\, X = 0\big) = p_{01}(T) \tag{1}$$

$$\Pr\big(Y = 0 \,\big|\, X = 1\big) = p_{10}(T) \tag{2}$$

$$\Pr\big(Y = \epsilon \,\big|\, X = 0\big) = q_0(T) \tag{3}$$

$$\Pr\big(Y = \epsilon \,\big|\, X = 1\big) = q_1(T) \tag{4}$$

$$\Pr\big(Y = \epsilon \,\big|\, X = \epsilon\big) = 1 \tag{5}$$

Equation (5) implies that a hard error always remains unchanged.

We label channels with $p_{01}(T) = p_{10}(T)$ and $q_0(T) = q_1(T)$ as symmetric channels and define the probability of soft error during time $T$ as

$$p(T) = p_{01}(T) = p_{10}(T) \tag{6}$$

with the probability of hard error during time $T$ as

$$q(T) = q_0(T) = q_1(T) \tag{7}$$

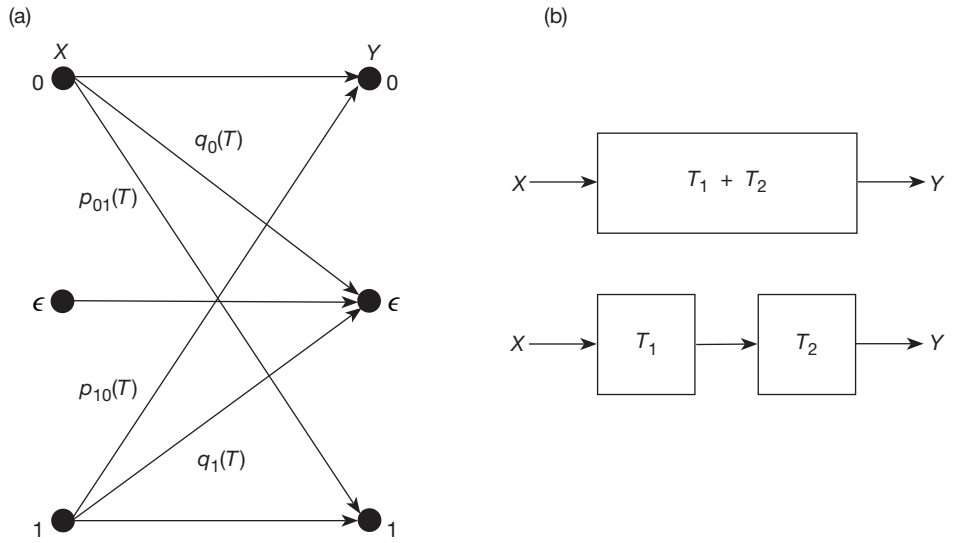


**Figure 1. Channel model of a one-bit memory without scrubbing: (a) channel model; (b) two equivalent channel models. Each transition probability is a function of time. Hard error is denoted as $\epsilon$. If $X = \epsilon$, the memory bit is not usable due to the hard error.**

and the probability of no error during time $T$ as $r(T)$. Thus,

$$p(T) = q(T) + r(T) = 1 \tag{8}$$

Now we will derive these probabilities by the following reasoning. If there is no scrubbing, a cascade of two channel models over successive time intervals $T_1$ and $T_2$ must be equivalent to one channel model over $T_1 + T_2$ as seen in Figure 1(b). Since we have a soft error if and only if there is only one soft error in $T_1$ or $T_2$ but no hard error, the soft error probability over the interval $T_1 + T_2$ must satisfy

$$p(T_1 + T_2) = p(T_1)r(T_2) + r(T_1)p(T_2) \tag{9}$$

Since we have a hard error if and only if there is a hard error in the first interval $T_1$; or no hard error in the first interval and a hard error in the second interval $T_2$, the hard error probability over the interval $T_1 + T_2$ must satisfy

$$q(T_1 + T_2) = q(T_1) + (1 - q(T_1))q(T_2) \tag{10}$$

We can obtain the following boundary conditions:

$$\begin{aligned} p(0) &= 0, & \lim_{T\to\infty} p(T = 0) \\ q(0) &= 0, & \lim_{T\to\infty} q(T = 1) \\ r(0) &= 1, & \lim_{T\to\infty} r(T = 0) \end{aligned} \tag{11}$$

since there is no error at the beginning and all the bits will eventually become hard errors.

Now we are ready to obtain $p(T)$, $q(T)$, and $r(T)$ by solving Equations (8), (9), and (10). One way is to solve differential equations by differentiating the equations with respect to the time variables and using the boundary conditions in Equation (11).

We obtained the following solutions for some nonnegative constants $\lambda$ and $\lambda_e$:

$$p(T) = \frac{e^{-\lambda_e T} - e^{-(2\lambda + \lambda_e)T}}{2} \tag{12}$$

$$q(T) = 1 - e^{-\lambda_e T} \tag{13}$$

$$r(T) = \frac{e^{-\lambda_e T} + e^{-(2\lambda + \lambda_e)T}}{2} \tag{14}$$

For a small time interval $T \ll \frac{1}{2\lambda + \lambda_e}$, we can approximate the soft error probability to $p(T) \approx \lambda T$ and the hard error probability $q(T) \approx \lambda_e T$. In this sense, $\lambda$ is called the *soft error rate* (the number of soft errors per bit per unit time), and $\lambda_e$ is called the *hard error rate* (the number of hard errors per bit per unit time). Strictly speaking, the unit for $\lambda$ and $\lambda_e$ is simply the inverse of time. However, error/bit/day is usually used for convenience. The soft error rates and the hard error rates are a measure of the radiation strength and its effects on memory systems.

In Figure 2(a), we see that the probability of hard error $q(T)$ increases monotonically, and the probability of no error $r(T)$ decreases monotonically. It might not be obvious why the probability of soft error $p(T)$ increases to the maximum and then decreases to zero. Figure 2(b) explains this behavior. In fact, the fraction of soft errors in the bits having no hard error increases monotonically from zero to half over time. The decrease of the probability of soft errors after the maximum is caused by the decrease of the probability of no hard error. Note that memory in our model can store no information at infinite time because $p(T)$ approaches 1/2, even if there is no hard error at all ($\lambda_e = 0$), which is of the same property as an infinite cascade of binary symmetric channels.

We will show an example in which a cascade of $L$ channel models with equal time intervals $T/L$ is equivalent to a channel model over time interval $T$. We will have a soft error over time interval $T$ if and only if we have an odd number of soft errors but no hard error. Therefore, we have the effective soft error probability over $T$

$$p_{\text{effective}}(T) = \sum_{j \text{ odd in } [0,L]} \binom{L}{j} \big(p(T/L)\big)^j \big(r(T/L)\big)^{L-j} \tag{15}$$

Since

$$\sum_{j=0}^{L} \binom{L}{j} \big(p(T/L)\big)^j \big(r(T/L)\big)^{L-j} = \big(r(T/L) + p(T/L)\big)^L \tag{16}$$

and

$$\sum_{j=0}^{L} \binom{L}{j} \big(-p(T/L)\big)^j \big(r(T/L)\big)^{L-j} = \big(r(T/L) - p(T/L)\big)^L \tag{17}$$

have the same terms for even powers of $p(T/L)$ and the terms of opposite signs for odd powers of $p(T/L)$,

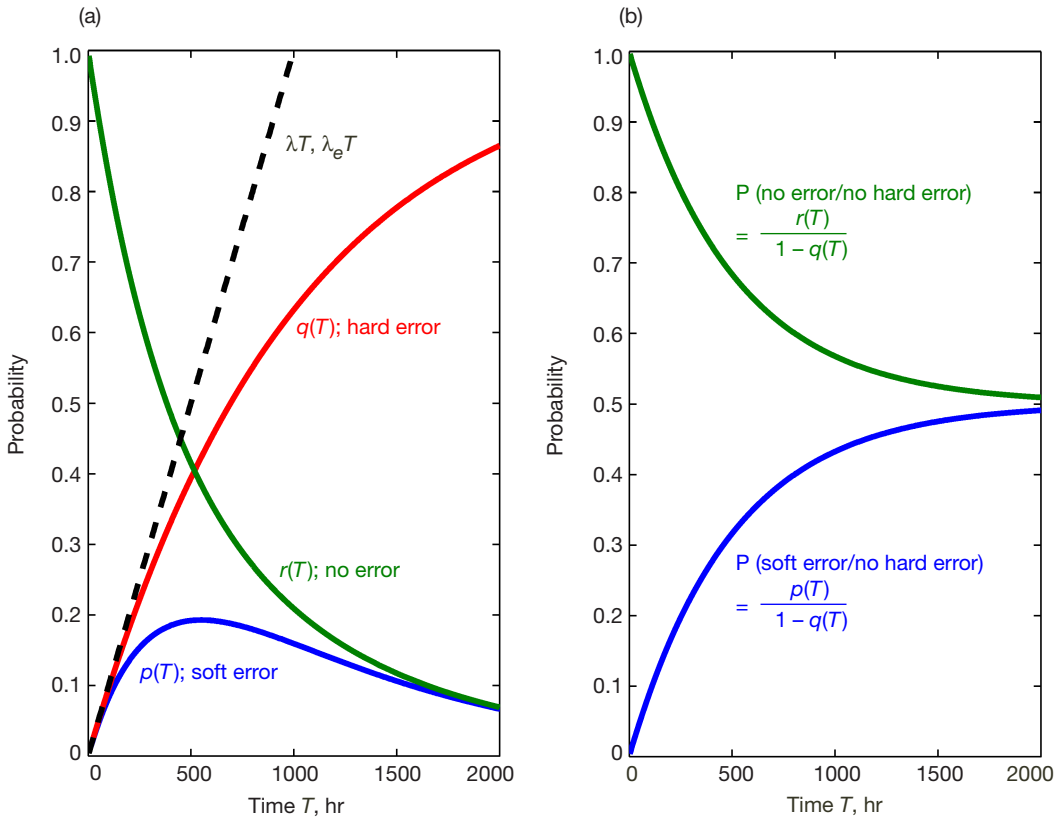


**Figure 2. Probabilities of soft error, hard error, and no error without scrubbing as functions of time when $\lambda = \lambda_e = 10^{-3}$ errors/bit/day: (a) probabilities; (b) conditional probabilities.**

$$p_{\text{effective}}(T) = \frac{1}{2}\left(\left(r(T/L) + p(T/L)\right)^L - \left(r(T/L) - p(T/L)\right)^L\right) \tag{18}$$

$$= \frac{1}{2}\left(\left(e^{-\lambda_e T/L}\right)^L - \left(e^{-(\lambda_e + 2\lambda)T/L}\right)^L\right) \tag{19}$$

$$= \frac{1}{2}\left(e^{-\lambda_e T} - e^{-(\lambda_e + 2\lambda)T}\right) \tag{20}$$

$$= p(T) \tag{21}$$

We have a hard error over $L$ stages if and only if we have a hard error at a stage $j$ but no hard error until that stage. Therefore, we have the effective hard error probability over $T$

$$q_{\text{effective}}(T) = \sum_{j=1}^{L} \left(1 - q(T/L)\right)^{j-1} q(T/L) \tag{22}$$

$$= 1 - \left(1 - q(T/L)\right)^L \tag{23}$$

$$= 1 - \left(e^{-\lambda_e T/L}\right)^L \tag{24}$$

$$= 1 - e^{-\lambda_e T} \tag{25}$$

$$= q(T) \tag{26}$$

Therefore, a cascade of $L$ channel models with equal time intervals $T/L$ is equivalent to a channel model over time interval $T$.

In this section, we defined a channel model that is consistent over any time interval.

### B. Channel Capacity Without Scrubbing

We will calculate the channel capacity without scrubbing (intermediate processing) in time $T$. For the symmetric cases, the channel capacity for interval $T$ is

$$C(T) = \max_{\Pr(X)} I(X;Y) \tag{27}$$

$$= 1 - q(T) + H(q(T)) - H(p(T),q(T)) \tag{28}$$

where the entropy functions are defined as

$$H(q(T)) \triangleq -q(T)\log_2 q(T) - (1-q(T))\log_2(1-q(T)) \tag{29}$$

and

$$H(p(T),q(T)) \triangleq -p(T)\log_2 p(T) - q(T)\log_2 q(T) - (1-p(T)-q(T))\log_2(1-p(T)-q(T)) \tag{30}$$

As a check for correctness, we see that when $p(T) = 0$, the capacity becomes $1 - q(T)$, which is the capacity of the binary erasure channel (BEC) with erasure probability $q(T)$. Similarly, if $q(T) = 0$, the capacity becomes $1 - H(p(T))$, which is the capacity of the binary symmetry channel (BSC) with crossover probability $p(T)$. For $0 < p(T) < \frac{1}{2}$ and $0 < q(T) < 1$, the channel model has smaller capacity than both the BEC with $q(T)$ and the BSC with $p(T)$.

It can be shown that

$$C/T = 1 - q(T) + H(q(T)) - H(p(T),q(T)) \tag{31}$$

$$= e^{\lambda_e T}\left(1 - H\left(\frac{1-e^{-2\lambda T}}{2}\right)\right) \tag{32}$$

$$= (1-q(T))\left(1 - H(p_{\lambda_e=0}(T))\right) \tag{33}$$

$$= C_{\mathrm{BEC}(q(T))} C_{\mathrm{BSC}(p_{\lambda_e=0}(T))} \tag{34}$$

where $p_{\lambda_e=0}(T)$ is the BSC crossover probability if the hard error probability were zero (i.e., $\lambda_e = 0$), $C_{\mathrm{BSC}(p_{\lambda_e=0}(T))}$ is the channel capacity of the BSC of crossover probability $p_{\lambda_e=0}(T)$, and $C_{\mathrm{BEC}(q(T))}$ is the channel capacity of the BEC of erasure probability of $q(T)$. In other words, the overall capacity is the product of channel capacities of a BEC and a BSC.

### C. Channel Capacity with Scrubbing

The channel capacity can be increased if we allow scrubbing over a time interval. We would expect that more frequent scrubbing will lead to a higher capacity. That is, if we use a shorter scrubbing interval, we can increase the code rate of the error-correcting code we use. Possible disadvantages of frequent scrubbing may include higher power consumption and shorter lifetime for flash memories due to wear-out effects.

The calculation of channel capacity with scrubbing is not straightforward. Although we may be able to calculate capacities asymptotically for infinite code length or infinite number of scrubbings, the capacity for finite code lengths and finite number of scrubbings remain as open questions in this article.

## III. Coding

### A. Review of RS and BCH Codes

Suppose that we have an $(n,k)$ BCH code that can correct up to $t$ bit errors per codeword if there is no bit erasure. The minimum distance of the BCH is $d_{\min} = 2t + 1$. As long as the number of bit errors (soft errors) $e$ and the number of bit erasures (hard errors) $f$ in a received word are bounded by

$$2e + f \leq 2t \tag{35}$$

the correct BCH codeword can be found by the decoder.

For the pseudodecoding of BCH codes, if Equation (35) is satisfied for a received word, we assume that all the errors and erasures are corrected and the corrected codeword is rewritten into memory during scrubbing. Otherwise, we declare a decoding failure and no scrubbing is performed. Note that once the number of accumulated hard errors (bit erasures) reaches $d_{\min} - 1$ in a codeword, no further errors (hard or soft) can be corrected.

Equation (35) also can be used for the pseudodecoding of an $(n,k)$ RS code over a Galois Field of size $2^m$ where we map $k$ information symbols into $n$ codeword symbols and each field element is represented by $m$ bits. This RS code can correct up to $t$ symbol errors if there are no symbol erasures and the minimum distance of the RS code is $d_{\min} = n - k + 1 = 2t + 1$.

An RS symbol is considered erased if one or more bits that comprise the symbol are erased. An RS symbol is considered erroneous when one or more bits that comprise the symbol is in error and no erasures occur in the symbol. Therefore, we obtain the probability of hard symbol error (symbol erasure) during a scrubbing interval as

$$q_s(T_s) = 1 - (1 - q(T_s))^m \tag{36}$$

and the probability of soft symbol error (erroneous symbol) as

$$p_s(T_s) = (1 - q(T_s))^m - (1 - p(T_s) - q(T_s))^m \tag{37}$$

The block error rates of RS codes can be calculated analytically by using the method provided in [4], if the probabilities of soft symbol error and hard symbol error are given. We also modified the method to analytically calculate the block error rates of BCH codes.

### B. LDPC Codes

Soft information can be generated using radiation parameters and the scrubbing interval. The log-likelihood ratio (LLR) $L_{ch}(y)$ for a memory output $y$ can be expressed in terms of the parameters in the channel model as follows:

$$L_{ch}(y) \triangleq \log \frac{\Pr\left(Y = y \,\middle|\, X = 0\right)}{\Pr\left(Y = y \,\middle|\, X = 1\right)} \tag{38}$$

$$= \begin{cases} \log \dfrac{p_{01}(T_s)}{1 - p_{10}(T_s) - q_1(T_s)}, & \text{if } y = 1 \\ \log \dfrac{1 - p_{01}(T_s) - q_0(T_s)}{p_{10}(T_s)}, & \text{if } y = 0 \\ \log \dfrac{q_0(T_s)}{q_1(T_s)}, & \text{if } y = \epsilon \end{cases} \tag{39}$$

For symmetric channels, the LLR from the memory output can be simplified as

$$L_{ch}(y) = \begin{cases} \pm \log \dfrac{p(T_s)}{r(T_s)}, & \text{if } y = 1(+), y = 0(-) \\ 0, & \text{if } y = \epsilon \end{cases} \tag{40}$$

These LLRs can be used not only in LDPC decoders but also in other soft decision decoders. Bit decisions are made based on the LLRs and the corrected bits can be used to update the soft errors but not the hard errors.

If we substitute $p(T_s)$ and $r(T_s)$ from Equation (12) for those in Equation (40), we obtain

$$L_{ch}(y) = \begin{cases} \pm \log \tanh(\lambda T_s), & \text{if } y = 1(+), y = 0(-) \\ 0, & \text{if } y = \epsilon \end{cases} \tag{41}$$

It is interesting to note that the hard error rate $\lambda_e$ is canceled out so that the LLR is independent of $\lambda_e$ for symmetric channels. In fact, it should not be a surprise since $p(T_s)/r(T_s) = \tanh(\lambda T_s)$ is the ratio between the points on the two curves in Figure 2(b) at $T = T_s$.

For LDPC decoding, either sum-product decoding or min-sum decoding can be used. The two methods are summarized in [10]. The min-sum decoding is an approximation of sum-product decoding and is less sensitive to the channel parameters in general.

### C. Turbo Product Codes

We describe a product code in Figure 3. The code comprises horizontal codewords and vertical codewords. The overall code can be seen as an $(n_1 n_2, k_1 k_2)$ code. In contrast to a general single $(n_1 n_2, k_1 k_2)$ code, the product code can be decoded using the decoders of the constituent $(n_1, k_1)$ and $(n_2, k_2)$ codes, which are much less complex compared to the single-code decoder, especially when $n_1$ and $n_2$ are large. If $n_1 = n_2$, only one short decoder is needed. While using low-complexity decoders is an advantage, a disadvantage is the requirement of additional memory to store extrinsic LLRs that are exchanged between the horizontal decoding and the vertical decoding. For example, for a 1-Mbit product code (i.e., $n_1 n_2 = 10^6$), 7-Mbit memory overhead is necessary if a 7-bit LLR implementation is used. In such a case, if a 2-Gbyte memory is used, the fraction of overhead for LLR storage will be $7 \times 10^6 / 16 \times 10^9 = 4.4 \times 10^{-4}$, which is small.

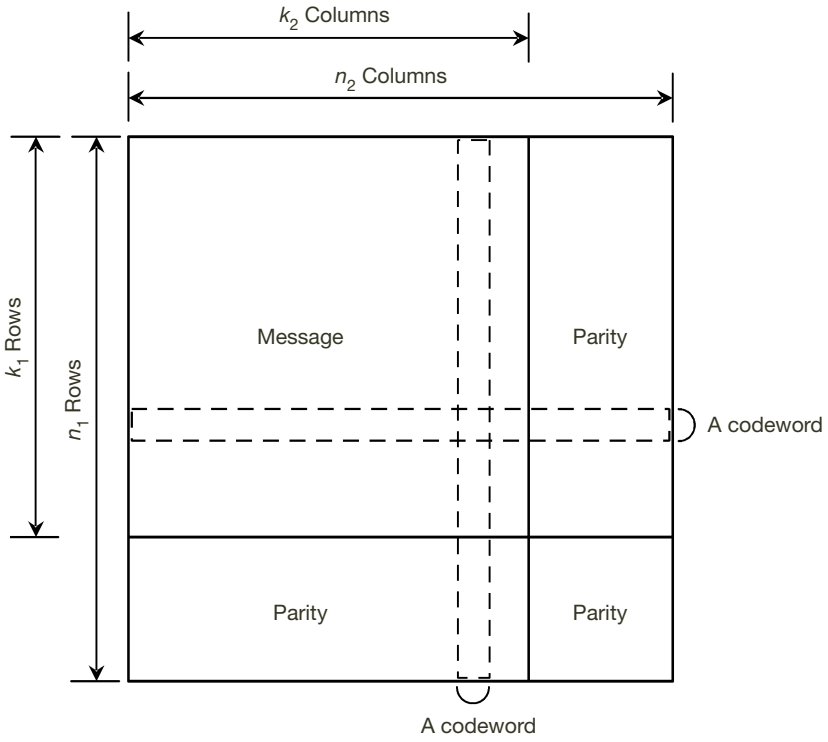


**Figure 3. Product codes.**

We illustrate the concept of turbo decoding of product codes in Figure 4. If the row (horizontal) codes and column (vertical) codes are the same, a common decoder can be shared between the row decoding and the column decoding. If throughput is a critical requirement, multiple decoders can be employed. The decoding algorithm is similar to that in [11], where each LDPC decoder provides extrinsic LLRs directly rather than posteriors.

We define one turbo iteration as (column decoding)-(row decoding) and two turbo iterations as (column decoding)-(row decoding)-(column decoding)-(row decoding).

## IV. Results

Figure 5 shows the block error rates of equivalent RS, BCH, and LDPC codes with a message length of 4096 bits and a code rate 8/9. This figure shows that LDPC code with sum-product decoding provides significant advantage over not only conventional RS or BCH codes but also the same LDPC code with min-sum decoding. Before we discuss the results further, we will describe the simulation setup.

The horizontal axis represents elapsed time and its unit is the number of scrubbing intervals. For example, 100 scrubbing intervals correspond to 100 hours when each scrubbing interval is 1 hour long. For the simulation, we used the probability of soft error and hard error as $4.167 \times 10^{-5}$. These soft and hard error probabilities do not change as long as the products $\lambda T_s$ and $\lambda_e T_s$ remain the same. One combination is $T_s = 1$ hour and $\lambda = \lambda_e = 10^{-3}$ errors/bit/day.

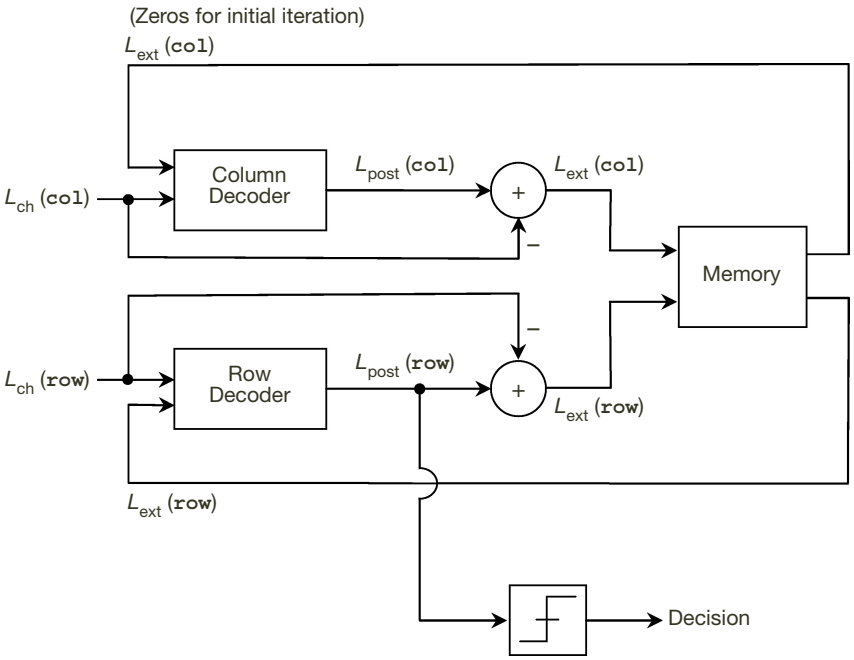


**Figure 4. Turbo decoding of product codes.**

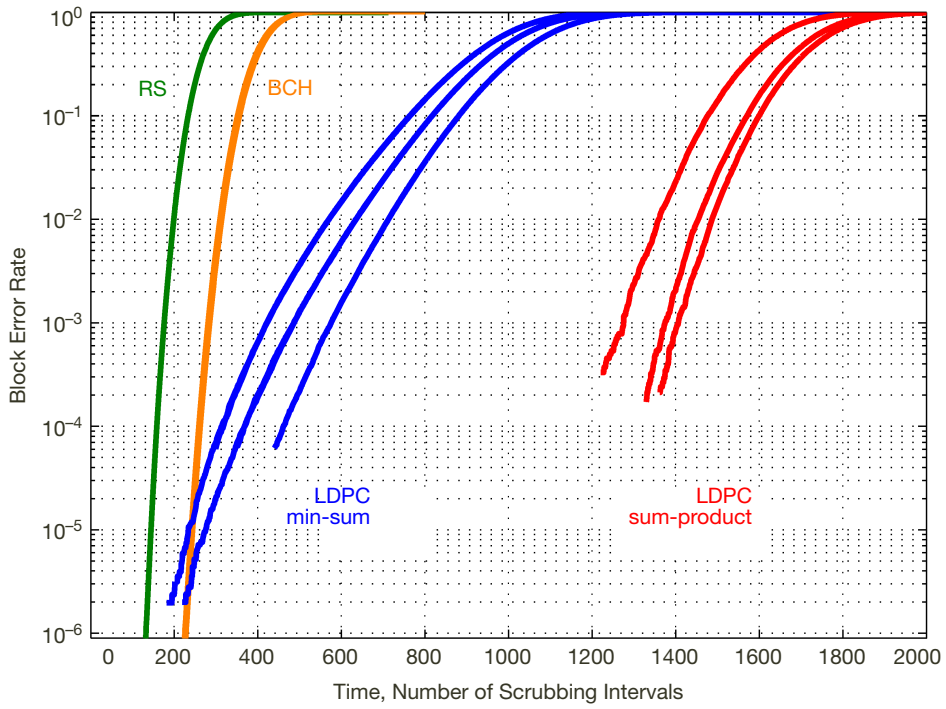


**Figure 5. Performance of a (4608, 4096) LDPC code; $p(T_s) = q(T_s) = 4.17 \times 10^{-5}$; $\lambda = \lambda_e = 10^{-3}$ error/bit/day and $T_s = 1/24$ day. The number of maximum iterations of LDPC decoder per scrubbing were 10, 20, and 40.**

Note that this radiation parameter translates into a harsher condition than that measured in space. Typically, $\lambda$ and $\lambda_e$ are in the range of $10^{-7}$ to $10^{-8}$ in space [6–8]. For $\lambda = \lambda_e = 10^{-7}$, the results in Figure 5 correspond to the case of $T_s = 10^4$ hours.

The vertical axis in Figure 5 denotes block error rates. A block error is declared when we obtain either a decoding failure or an incorrect codeword estimate at the decoder output. We do not show results corresponding to fewer than 5 block errors.

The RS code is a shortened (462, 410) code over GF($2^{10}$) that can correct up to 26 soft symbol errors ($t = 26$ symbols, $d_{\min} = 53$ symbols). Two BCH codes with code rates slightly above 8/9 and below 8/9 are plotted in Figure 5 since the BCH code with code rate exactly 8/9 and the message length 4096 bits does not exist. The lines for the two codes are too close to each other to be distinguishable in Figure 5. One code is a (4603, 4096) code shortened from the (8191, 7684) BCH code that can correct up to 39 soft errors in a codeword ($t = 39$ bits, $d_{\min} = 79$ bits). Another code is a (4616, 4096) code shortened from (8191, 7671) code that can correct up to 40 soft errors in a codeword ($t = 40$ bits, $d_{\min} = 81$ bits). The block error rates obtained by the analytical method [4] for the RS code and our modification for the BCH codes were identical to our simulation results.

The LDPC code used in Figure 5 is a (4608, 4096) code whose parity check matrix was generated by the progressive edge growth (PEG) method [12,13]. The column weight of the parity check matrix is 5 and the girth of the bipartite graph is 6. The minimum distance of this LDPC code is unknown, which is typical for most for LDPC codes. In general, computing the minimum distance of a binary linear code is an NP-hard problem [14]. RS codes and BCH codes have nice algebraic constructions in which the desired minimum distances are defined by design. For each of min-sum decoding and sum-product decoding, the number of maximum iterations for the three lines is 10, 20, and 40 from top to bottom.

We see that LDPC code with sum-product decoding provides significant gains over RS or BCH codes. Meanwhile, the advantage of LDPC code with min-sum decoding over RS and BCH codes decreases as target block error rates decrease and the advantage even disappears when the curves cross each other in low block error rates. The sum-product curves could cross the RS or BCH curves in the low block error rate region. However, if we assume that the slopes of the curves are maintained, the crossing point is expected to be at extremely low block error rates so that the LDPC code with sum-product decoding provides an advantage over RS or BCH codes for practical applications.

The performance gain of sum-product decoding over min-sum decoding is related to the importance of the usage of the radiation parameters in the LLR determination in Equation (38).

Figure 6 shows the block error rates of equivalent RS, BCH, and LDPC codes with a message length of 2048 bits and a code rate 8/9. The simulation setup is the same as in Figure 5 except for the length of the error-correcting codes. The RS code is a shortened (231, 205) code over GF($2^{10}$) that can correct up to 13 soft symbol errors ($t = 13$ symbols, $d_{\min} = 27$ symbols). One BCH code (left curve) is a (2300, 2048) code shortened from the (4095, 3843)

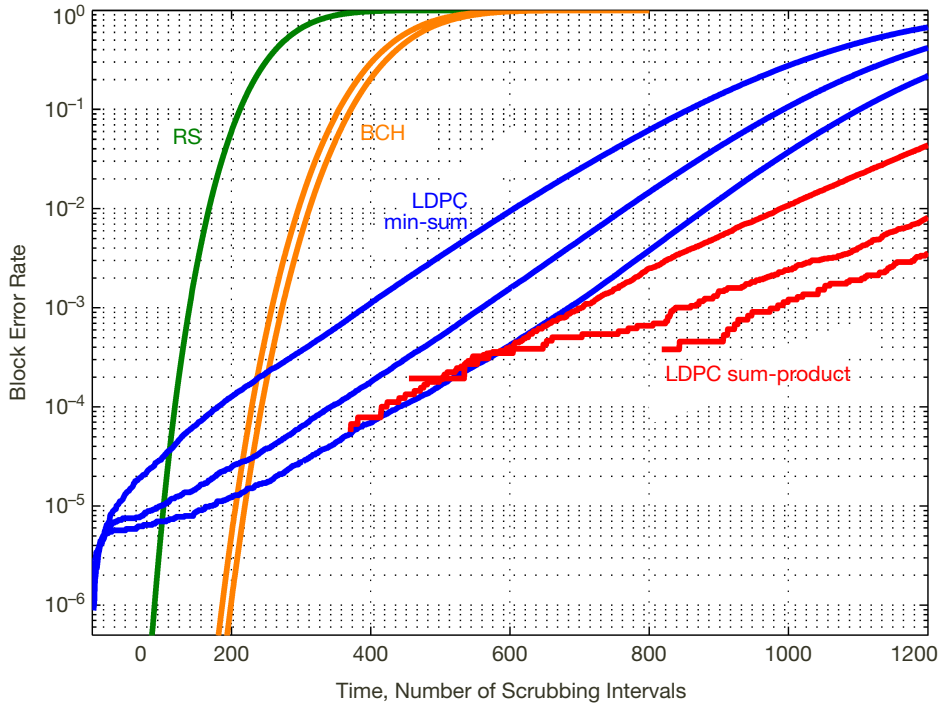


**Figure 6. Performance of a (2304, 2048) LDPC code; $p(T_s) = q(T_s) = 4.17 \times 10^{-5}$; $\lambda = \lambda_e = 10^{-3}$ error/bit/day and $T_s$ = 1/24 day. The number of maximum iterations of LDPC decoder per scrubbing were 10, 20, and 40.**

BCH code that can correct up to 21 soft errors in a codeword ($t = 21$ bits, $d_{\min} = 43$ bits). The other BCH code (right curve) is a (2312, 2048) code shortened from the (4095, 3831) BCH code that can correct up to 22 soft errors in a codeword ($t = 22$ bits, $d_{\min} = 45$ bits). The LDPC code is a (2304, 2048) PEG code with the column weight 3 and girth 6. For each of min-sum decoding and sum-product decoding, the number of maximum iterations for the three lines is 10, 20, and 40 from top to bottom.

We observe similar behaviors of RS code, BCH code, and LDPC code with min-sum decoding as in Figure 5. Interestingly, the advantage of the LDPC with sum-product decoding is diminished significantly, so that even the sum-product decoding can be outperformed by BCH codes at block error rates below $10^{-5}$ if we assume that the slope of the curves is maintained. Therefore, for these 2-kbit codes, using LDPC codes can provide gains only if target block error rates are above about $10^{-5}$.

Figure 7 shows the block error rates of equivalent RS, BCH, and LDPC codes that can contain 1024 bits of message at code rate 8/9. The simulation setup is also the same as in Figure 5 except for the length of the error-correcting codes. The RS code is a shortened (144, 128) code over GF($2^8$) that can correct up to 8 soft symbol errors ($t = 8$ symbols, $d_{\min} = 17$ symbols). One BCH code (left curve) is a (1156, 1024) code shortened from the (2047, 1915) BCH code that can correct up to 12 soft errors in a codeword ($t = 12$ bits, $d_{\min} = 25$ bits). The other BCH code (right curve) is a (1145, 1024) code shortened from the (2047, 1926) BCH code that can correct up to 11 soft errors in a codeword ($t = 11$ bits, $d_{\min} = 23$ bits). The LDPC code is a (1152, 1024) PEG code with column weight 3 and

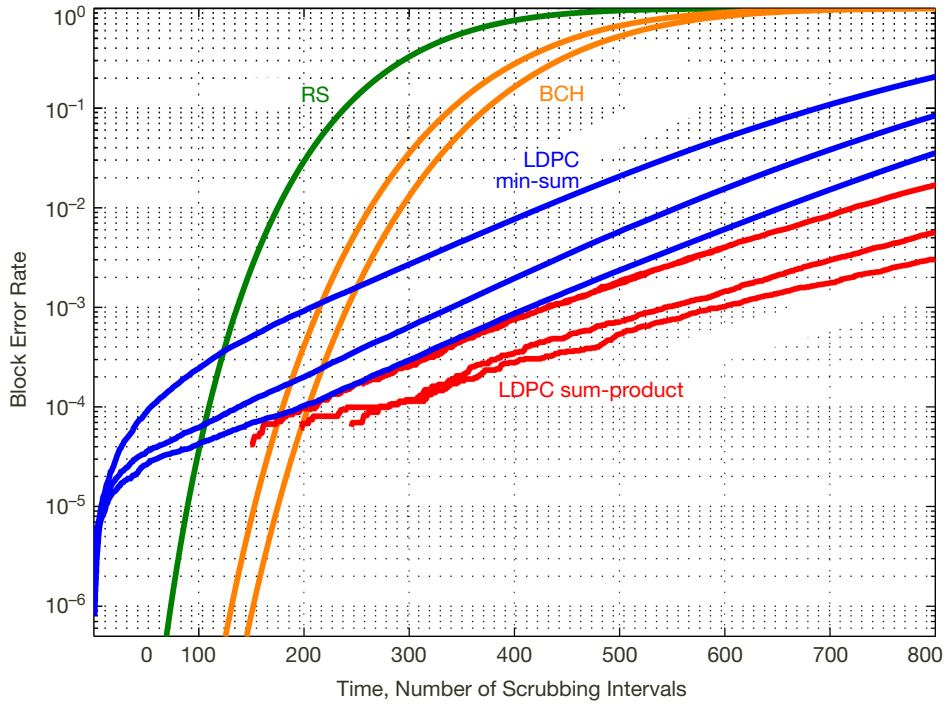


**Figure 7. Performance of a (1152, 1024) LDPC code; $p(T_s) = q(T_s) = 4.17 \times 10^{-5}$; $\lambda = \lambda_e = 10^{-3}$ error/bit/day and $T_s$ = 1/24 day. The number of maximum iterations of LDPC decoder per scrubbing were 10, 20, and 40.**

girth 6. For each of min-sum decoding and sum-product decoding, the number of maximum iterations for the three lines is 10, 20, and 40 from top to bottom.

We can observe similar behaviors of RS code, BCH code, and LDPC code as in Figure 6. The gain of sum-product decoding over min-sum decoding becomes even smaller than in Figure 6. Comparing Figures 5, 6, and 7, the gain of sum-product decoding over min-sum decoding increases as the code length increases at a fixed code rate. In particular, the gain from the 2-kbit code to the 4-kbit code is very large, whereas the gain from the 1-kbit code to the 2-kbit code is small.

JPL has designed structured LDPC codes based on protographs and circulants [15,16]. This construction enables high-speed decoder implementations because the component protographs that are the building blocks to the bigger code graph can be decoded in parallel. The structure of the protograph then determines the threshold and error floor of the overall code. Divsalar et al. [17] recognized that a protograph described by simple accumulate and repeat operators can yield codes with sharp waterfalls and low error floors. We plot the performance of the rate 4/5 information block size 1024-bit accumulate repeat-by-4 jagged accumulate (AR4JA) LDPC code in Figure 8 and compare the performance to an equivalent rate and length RS and BCH codes: (160, 128) RS code over GF($2^8$), (1277, 1024) BCH code shortened from (2047, 1794) BCH code that can correct up to 23 soft errors in a codeword ($t$ = 23 bits, $d_{\min}$ = 47 bits), and (1288, 1024) BCH code shortened from the (2047, 1783) BCH code that can correct up to 24 soft errors in a codeword ($t$ = 24 bits, $d_{\min}$ = 49 bits). As with the PEG LDPC code, the AR4JA code outperforms the RS and BCH codes for high block error rates.

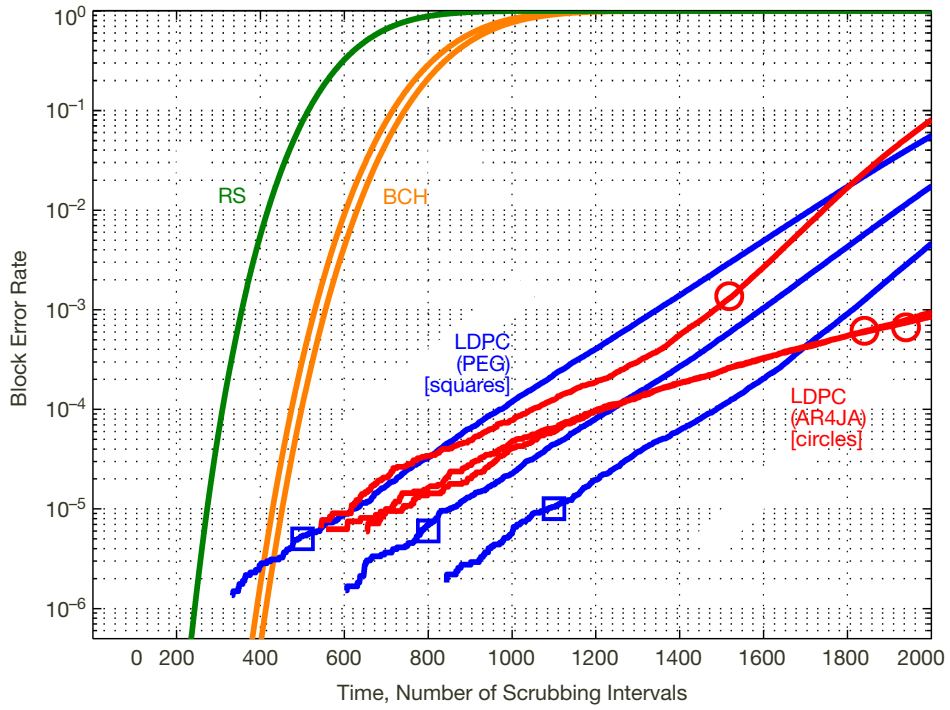


**Figure 8. Performance of two (1280, 1024) LDPC codes (min-sum decodindg): an accumulate-repeat-by-4-jagged-accumulate (AR4JA) LDPC code anda PEG LDPC code. All codes are rate-4/5; $p(T_s) = q(T_s) = 4.17 \times 10^{-5}$; $\lambda = \lambda_e = 10^{-3}$ error/bit/day and $T_s$ = 1/24 day. The number of maximum iterations of LDPC decoder per scrubbing were 10, 20, and 40.**

We also simulated decoding of turbo product codes (TPC) on the same channel and plot the performance curves in Figure 9. Again, we set both the soft and hard error rates to $10^{-3}$ error/bit/day and set the scrubbing interval to one hour. All codes in the figure have rate-$(8/9)^2$. The (1296, 1024) LDPC code is a PEG code with column weight 5 and girth 6 and decoded by running two iterations. The $(1152, 1024)^2$ TPC comprises two constituent (1152, 1024) LDPC codes as the row and column code. We denote the number of LDPC decoding iteration by "L" and the number of turbo iterations by "T," so (L1,T2) represents one LDPC decoding and two turbo decoding iterations. We simulated 50 TPC codewords. Again, the fluctuation at low bit error rate (BER) is due to lack of averaging over enough TPC codewords.

We see that increasing the number of turbo iterations is more effective than increasing the number of LDPC iterations when turbo decoding product codes. With only two LDPC and two turbo decoding iterations, TPC began to outperform the (1296, 1024) LDPC code with the same code rate. Effectively, TPC is a longer code and its performance shows.

## V. Summary

To protect against radiation-induced errors in space, memory systems either store the same information repeatedly in different memory locations or use a conventional ECC scheme such as SECDED Hamming or RS codes. We developed a simple channel that models single

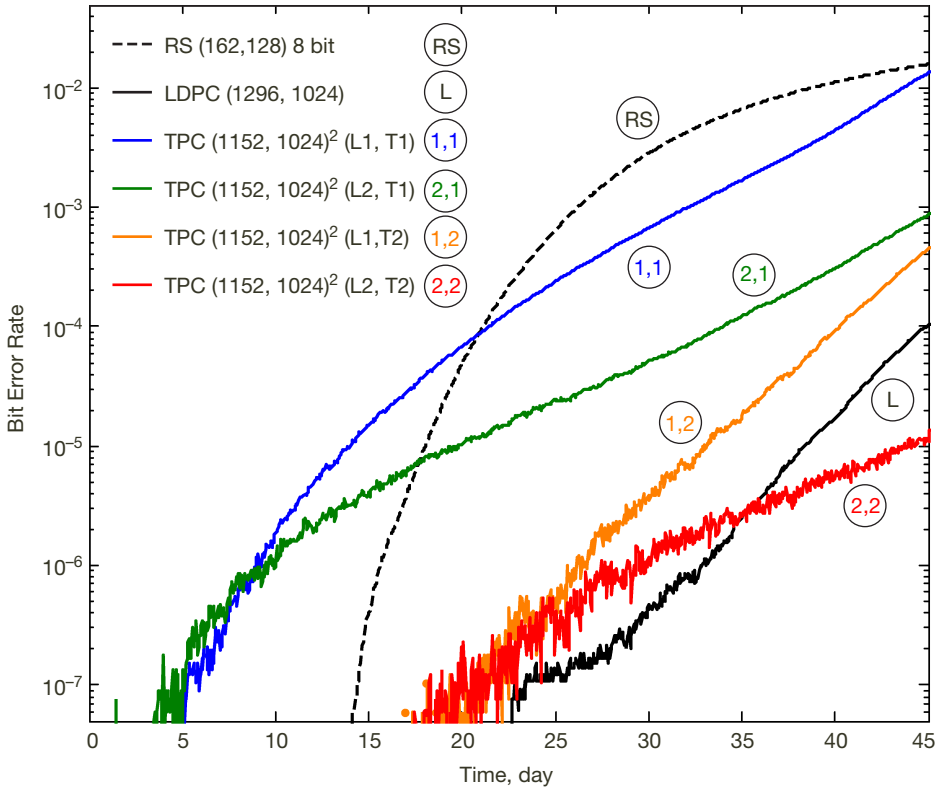


**Figure 9. Performance of product codes; $\lambda = \lambda_e = 10^{-3}$; $T_S$ = 1/24 day. All codes are rate-$(8/9)^2$.**

bit errors due to radiation, assuming bit errors occur independently from bit-to-bit. We discussed the capacity of this channel. We showed that modern LDPC codes can be used in place of conventional RS and BCH codes to improve the radiation tolerance of memory modules. Instead of decoding hard (i.e., 0 or 1) bits, LDPC decoding uses soft information provided by the channel. We showed how to compute soft symbol reliabilities on our channel for input to soft-decision LDPC decoders. We considered two LDPC code constructions: one generated by progressive edge growth and the other is based on protographs and compared their performances to equivalent rate and length RS and BCH codes. To obtain an even stronger code, we looked at two-dimension product codes that have LDPC component row and column codes. The simulation results suggest that LDPC codes can extend the lifetime of memory systems over equivalent BCH or RS codes at relevant target block error rates in a radiation environment.

## References

[1] R. M. Goodman and M. Sayano, "The Reliability of Semiconductor RAM Memories with On-Chip Error-Correction Coding," *IEEE Transactions on Information Theory*, vol. 37, no. 3, pp. 884–896, May 1991.

[2] A. M. Saleh, J. J. Serrano, and J. H. Patel, "Reliability of Scrubbing Recovery-Techniques for Memory Systems," *IEEE Transactions on Reliability*, vol. 39, no. 1, pp. 114–122, April 1990.

[3] P. P. Shirvani, N. R. Saxena, and E. J. McCluskey, "Software-Implemented EDAC Protection Against SEUs," *IEEE Transactions on Reliability*, vol. 49, no. 3, pp. 273–284, September 2000.

[4] G. C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Data Integrity Evaluations of Reed-Solomon Codes for Storage Systems," *Proceedings of the 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT04)*, Cannes, France, October 2004.

[5] G. C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Fault-Tolerant Solid-State Mass Memory for Space Applications," *IEEE Transactions on Aerospace Electronic Systems*, vol. 41, no. 4, pp. 1353–1372, October 2005.

[6] H. Kaneko, "Error Control Coding for Semiconductor Memory Systems in the Space Radiation Environment," *Proceedings of the 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)*, Monterey, California, October 2005.

[7] T. C. MacLeod, R. Sayyah, W. H. Sims, K. A. Varnavas, and F. D. Ho, "Satellite Test of Radiation Impact on Ramtron 512K FRAM," *10th Annual Non-Volatile Memory Technology Symposium (NVMTS'09)*, Portland, Oregon, October 2009.

[8] D. N. Nguyen, "Radiation Effects on NAND Flash Memories," *10th Annual Non-Volatile Memory Technology Symposium (NVMTS'09)*, Portland, Oregon, October 2009.

[9] S. Jeon, E. Hwang, B. V. K. Vijaya Kumar, and M. K. Cheng, "Investigation of Memory Protection Using Low-Density Parity-Check (LDPC) Codes," *10th Annual Non-Volatile Memory Technology Symposium (NVMTS'09)*, Portland, Oregon, October 2009.

[10] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, February 2001.

[11] J. Li, K. R. Narayanan, E. Kurtas, and C. N. Georghiades, "On the Performance of High-Rate TPC/SPC Codes and LDPC Codes Over Partial Response Channels," *IEEE Transactions on Communications*, vol. 50, no. 5, pp. 723–734, May 2002.

[12] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Progressive Edge-Growth Tanner Graphs," *Proceedings of IEEE Global Communications Conference 2001 (IEEE GLOBECOM'01)*, vol. 2, San Antonio, Texas, pp. 995–1001, November 2001.

[13] Z. Li and B. V. K. Vijaya Kumar, "A Class of Good Quasi-Cyclic Low-Density Parity Check Codes Based on Progressive Edge Growth Graph," *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems, and Computers*, vol. 2, Pacific Grove, California, pp. 1990–1994, November 2004.

[14] A. Vardy, "The Intractability of Computing the Minimum Distance of a Code," *IEEE Transactions on Information Theory*, vol. 43, no. 6, pp. 1757–1766, November 1997.

[15] J. Thorpe, "Low-Density Parity-Check Codes Constructed from Protographs," *The Interplanetary Network Progress Report*, vol. 42-154, Jet Propulsion Laboratory, Pasadena, California, pp. 1–7, August 15, 2003.
http://ipnpr.jpl.nasa.gov/progress_report/42-154/154C.pdf

[16] 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, Special Issue on Technical Advances in Deep-Space Communications and Tracking: part 2, November 2007.

[17] D. Divsalar, S. Dolinar, C. Jones, and J. Thorpe, "Construction of Protograph LDPC Codes with Minimum Distance Linearly Growing with Block Size," *Proceedings of IEEE Global Communications Conference 2005 (IEEE GLOBECOM'05)*, St. Louis, Missouri, November 2005.