Early-Stopped Approach and Analysis for the Berlekamp-Massey Algorithm

Hong-fu Chou
Interdisciplinary Centre for Security, Reliability, and Trust, University of Luxembourg
Email: exhan100chou@gmail.com

***Abstract*—BCH codes are being widely used in commercial NAND flash controllers, and the decoding algorithm based on the Berlekamp-Massey (BM) algorithm is a classic solution for solving the key equation used for error correction. The latency of BM decoding is the bottleneck of the Bose-Chaudhuri-Hocquenghem (BCH) decoder when correcting a high number of bit errors. However, the flash memory has an error distribution that degrades with usage: few errors occur in the new memory and a low number of errors occur within a code block. With usage, the system performance degrades and BM decoding needs t iterations in order to correct a larger number t of errors. In an attempt to improve the system performance for high speed applications, early termination of the BM decoding is necessary to overcome this degradation. In this paper, a practical solution for early termination checking for BM algorithm is provided. The analysis of proposed method is presented by means of considering the weight distribution of BCH code and deriving the probability of malfunction as the event of undetectable error. The proposed method is presented to be effective by the numerical results and the probability of malfunction for the proposed method is lower than 10^{-26}. As a result,the FPGA testing on a USB device validate the reliability of the proposed method for applying to a commercial product.**

I. Introduction

Flash memory [1] [2] performs as the main non-volatile storage device, and the flash interface unit is applied for system on chip (SoC) products. The market demand for non-volatile flash memory is increasing, since the development of new applications that are not reliant on heavy hard disks has been rapidly growing recent years. Flash memory provides a low power solution for storage systems and, it is worth mentioning the small size and the light form factor are the essential properties for this type of storage. The flash interface unit [3] provides basic flash commands which can be used by the main central processing unit(CPU) to access data from the flash memory. It is assumed that the flash memory is non-removable, since the flash memory is used to initiate the boot process based on information from the firmware.

The flash memory plays an important role for the storage device to execute the tasks to be performed by the main CPU. The tasks are literally to read and write files and are identical to any generic file system. The flash interface unit has mainly provided a reliable component for graphics and multimedia processors and have been applied to digital televisions, car navigation systems, and mobile applications. To support multimedia applications, flash interface units have been optimized for large block read and write, as presented in [4]. To minimize the main CPU interaction, the flash interface unit supports direct memory access (DMA) [5] when transferring from the flash memory to the system DRAM memory.

In the SoC applications, all of the boot information is generally stored in the flash memory. The flash memory includes a number of partitions for the boot loader code and the flash file system is created in the flash memory. In [6], the DMA interacts with the error control coding (ECC) block, which provides two main purposes. The first is to generate the ECC bytes and program in the spare area, and the second is to correct the data in the data buffer. Consequently, the ECC engine is a critical issue regarding system performance. The chip area is dominated by the ECC decoder, comprising a high percentage of the flash controller.

The Bose-Chaudhuri-Hocquenghem (BCH) code has become the ultimate solution for the ECC engine in recent years. In coding theory, the BCH codes form a class of cyclic error-correcting codes that are constructed using finite fields. The decoding algorithm is based on a feasible implementation where the Berlekamp-Massey (BM) algorithm [7] has been widely selected in typical examples. The complexity of the decoding is competitive with respect to the BM properties of the linear feedback shift register. However, system latency degrades by t iterations of BM decoding, and common applications require high error-correcting capability. The long decoding time has become a bottleneck in the system performance while using BM decoding. The error distribution for flash memory shows that few errors at the beginning of its usage and the low number of errors dominates the majority of the probability that will occur within a code block. In order to overcome this degradation, early termination of BM decoding is necessary to improve the system performance for high speed applications. In [8], the author adopt a restricted Gaussian elimination on the Hankel structured augmented syndrome matrix to reinterpret an early-stopped version of the Berlekamp-Massey algorithm. However, this approach has revealed a general presentation to the BM algorithm which is quite difficult to directly implement on hardware description language. According to the thread of [9], the author present a feasible approach for early termination but the investigation of malfunction probability was present in [10].

In this paper, we present a feasible decoding strategy where the BM decoding terminates after less than t iterations based on the detection of the number of error bits so as to reduce redundant latency. Consequently, an early termination mechanism needs to be developed, and the probability that a detection error will occur must be evaluated to ensure the

reliability of the new approach. In this paper, we propose an early-stopped approach for BM decoding by observing certain conditions while performing decoding iterations. In Section II, we provide the early-stopped version of BM decoding, together with an analysis of the probability of undetected errors. In Section III, the numerical results are presented. Conclusions are presented in Section IV.

II. Early stopped approach based on the view of discrepancy for the BM algorithm

In coding theory, BCH codes [11] [12] are constructed using polynomials over a finite field (also called the Galois field and is denoted as GF(q)). One of the key features of BCH codes is that, during code design, there is precise control over the number of symbol errors that are correctable by the code. In particular, it is possible to design binary BCH codes that can correct multiple bit errors under a correction capability of t bits. Another advantage of BCH codes is the ease with which they can be decoded, namely, via an algebraic method known as syndrome decoding. This simplifies the design of the decoder for these codes, using small low-power electronic hardware.

BCH codes are used in applications such as satellite communications, compact disc players, DVDs, disk drives, and solid-state drives, etc.

There are many algorithms for decoding BCH codes. The most common follow this general outline:

1. Calculate the syndromes for the received vector
2. Determine the number of errors v and the error locator polynomial $N(x)$ from the syndromes
3. Calculate the roots of the error location polynomial to determine the error locations X_i
4. Calculate the error values at those error locations
5. Correct the errors

During some of these steps, the decoding algorithm may determine that the received vector contains too many errors and cannot be corrected. For example, if the number of errors is greater than correction capability, then the correction would fail. In a truncated (not primitive) code, an error location may be out of range. If the received vector has more errors than the code can correct, the decoder may unknowingly produce an apparently valid message that is not the one that was sent.

In order to determine any possible solutions to shorten the BM decoding process, based on Chen's result in [9], we classify the solutions in two statements as follows.

Statement (1):

For the u-th iteration of the BM algorithm, the discrepancy at iteration u is presented as d_u and any discrepancies in the next (t-l_u-1)/2 steps of the iteration are zero.

Statement (2):

If the number of errors in the received polynomials is v, only $(t+v)/2$ steps of the iteration are needed in order to determine the error-location polynomials.

The discrepancies in certain iterations equal to zero, as shown in Statement (1) represents the detection capability reach in certain level of l_u iterations, i.e. $l_u = v$, where v is the number of errors hypothesized by our proposed approach. Let a code ζ have minimum distance $d \geq 2t+1$ and consider that ζ^{t+v+z} denotes a $(t+v+z-1)/2$-error-correcting BCH code. According to [10], we present the investigation as follows. The Hamming distance for the received codeword r and the transmitted codeword c is presented as $d(r,c) = i, i < t$, where $c \in \zeta^{t+v+z}$ and z are the number of consecutive zero discrepancies. For example, r is a 4-bit zero vector and c is a 4-bit one vector. the value of $d(r,c)$ is equal to 4. Since error pattern e defects the codeword c, it can also be presented as $r = c + e$ and $d(r,c) = d(e,c)$. We consider the case where we assume $l_u = v$, which indicates that early termination is successfully achieved using our proposed approach. Otherwise, a malfunction occurs when the location of the error pattern is beyond the detection capability at l_u iteration which means the case of $v + z < l_u$. Furthermore, a non-zero discrepancy occurs after $v + z$ iterations and the codeword $c \in \zeta^{t+v+z} - \zeta$. The proposed approach fails to provide a correct prediction of the number of errors. The probability of malfunction is given as follows.

$$\begin{aligned} P_{mf} &= p[v + z < l_u] \\ &= \sum_{i=0}^{t} P[d(e,c) = i | c \in \zeta^{t+i+z} - \zeta] \\ &= \sum_{i=0}^{t} P[d(r,c) = i | c \in \zeta^{t+i+z}] - P[d(r,c) = i | c \in \zeta] \end{aligned} \quad (1)$$

According to [6], the probability that an undetected error will occur can be bound by the assumption of a long codeword length and m is equal to the message length,

$$\begin{aligned} P_e &= \sum_{i=0}^{t} P[d(e,c) = i | c \in \zeta^t - 0] \\ &\cong 2^{-mt} \sum_{s=0}^{t} \binom{n}{s} \sum_{h=t+1}^{n} \binom{n}{h} \varepsilon^h (1-\varepsilon)^{n-h} \end{aligned} \quad (2)$$

We further extend the probability of an error pattern is given by [13] and [6]

$$P[d(e,c) = i | c \in \zeta^d] \cong \sum_{h=(d+1)/2}^{n} \binom{n}{h} \varepsilon^h (1-\varepsilon)^{n-h} \quad (3)$$

Substituting this into (1), the probability of malfunction is

$$\begin{aligned} P_{mf} \cong 2^{-mt} [& \sum_{s=0}^{t} \binom{n}{s} \sum_{h=(s+t+z+1)/2}^{n} \binom{n}{h} \varepsilon^h (1-\varepsilon)^{n-h} \\ & - \sum_{s=0}^{t} \binom{n}{s} \sum_{h=t+1}^{n} \binom{n}{h} \varepsilon^h (1-\varepsilon)^{n-h}] \end{aligned} \quad (4)$$

Based on the concept of observing zero discrepancies during BM iteration, we illustrate the proposed early-stopped checking strategy, which is described below. The proposed method

is denoted as the early-stopped(ES) version, and we provide three different versions. For BM decoding of the j-th iteration, we observe the following discrepancy based on the proposed method. We denote that δ_{max} represents the maximum error location degree of the BM algorithm .

ES version 1:

Beginning from $j = 4$ in the BM algorithm, verify the following steps:
1. Check Case A: t+ $\delta_{max}/2$ = j
2. Check Case B: d_j, d_{j-1}, d_{j-2} and d_{j-3} are all zero.
3. If Case A and Case B are satisfied, terminate the BM decoding. Otherwise, proceed to the next BM iteration and return to Step 1.

ES version 2:

Beginning from $j = 4$ in the BM algorithm, verify the following steps:
1. Check Case A: $t + \delta_{max}/2$ = j
2. Check Case B: d_j, d_{j-1}, d_{j-2}, d_{j-3}, d_{j-4}, d_{j-5} are all zero.
3. If Case A and Case B are satisfied, terminate the BM decoding. Otherwise, proceed to the next BM iteration and return to Step 1.

ES version 3:

Beginning from $j = \kappa + 1$ in the BM algorithm, verify the following steps:
1. Check the Case A: d_j, d_{j-1}, ..., $d_{j-\kappa}$ are all zero.
2. If Case A is satisfied, terminate the BM decoding. Otherwise, proceed to the next BM iteration and verify Step 1.
κ is set to 1, 2, 3, 4 or 6 before simulation.

III. Numerical result

Previous approaches were applied the performing t-iteration algorithm and limited to the small t condition based on the concern of hardware complexity increasing with larger t. The proposed early stopped technique has the capability to reduce the decoding latency. For example, the case of t which is equal to 72 leads to a huge cost of the area to implement the BCH decoder and the decoding latency of BM decoding degrade the system performance of the DMA accessing the flash memory. However, the previous researcher did not consider the early stopped method to the practical application for the storage system. According the authors of [9] and [10], the probability of undetected errors occurring in binary linear codes can be simplified and quantified if the weight distribution of the code is binomial-like. The authors obtained bounds on the probability of undetected errors in binary primitive BCH codes by applying the result to the code, and showed that the bounds are quantified by the deviation factor of the true weight distribution from the binomial-like weight distribution. The authors provided a close upper bound on the probability of undetected errors. This approach presents a promising prediction for us to investigate how a long primitive BCH code can be secure to applying early stopped approach in a NAND flash system.

The prediction of undetected errors can be evaluated via the simulation of ML decoding on the short BCH code to determine how close it is to the prediction, and to identify what actually happens while decoding process is executing. Below, we provide an example that reveals how the simulation processes. First, we consider a BCH code with a length that is equal to 31 in $GF(2^5)$, and that can correct $t = 4$, the length of the codeword is 15, which has an outcome of 2^{16} codewords. During the decoding of the received codeword used to compute the discrepancy, we consider the following case. If we observe

TABLE I: A case that illustrates early termination checking

Discrepancy	> 0	0	0	d’
BM iteration	1	2	3	4

that the number of discrepancies is consecutively zero, we can compute the probability of a failure event occurring if d’ is equal to non-zero. We denote the conditional probability as Po. A failure event can cause the proposed method to fail to decode a correct codeword. The result is shown below. The failure rate is illustrated based on the ML decoding for a certain degree of the non-zero discrepancy during each iteration. In Fig 1, it can be observed that Po has the lower bound of 10^{-4}. This result implies that the failure rate could approach an extremely small value, while the code length becomes longer. The probability of undetected errors in the

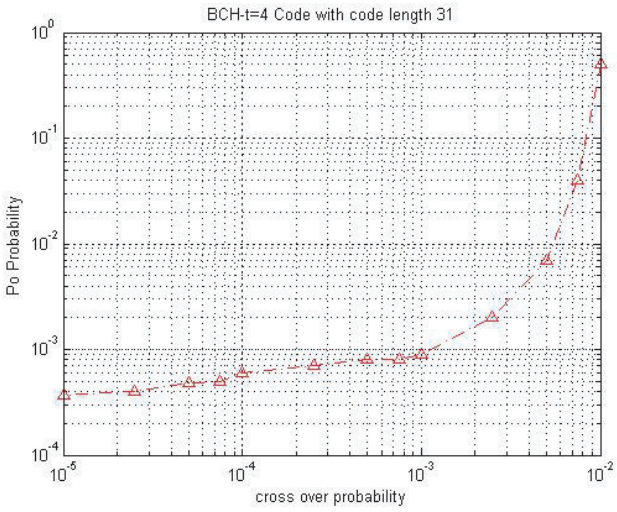

Fig. 1: The probability of undetected errors during early termination checking using $GF(2^5)$ BCH code t=4.

proposed ES version can be calculated using equation (4). To approach the upper bound of the probability of malfunction, a BCH code with a length 1023 and $t = 17$ is presented as an example to reveal the effectiveness of equation (4). For ES versions 3 with $\kappa = 6$, the probability of undetected errors is calculated as 5.24051×10^{-26} over the cross over probability at 10^{-2}. In Fig 2, it can be shown an example that ES version 3 provides a reliable result for early termination checking by observing that the number of discrepancies is

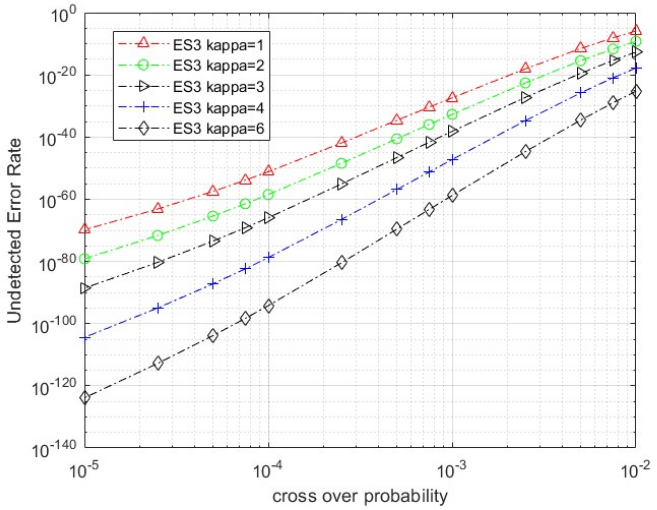

Fig. 2: The probability of undetected errors during early termination checking for ES version 3 using $GF(2^{10})$ BCH code t=17.

consecutively zeros. For practical applications, the proposed ES version 3 can prevent decoding failure over the firmware and decoder commuting period. As shown in equation (4), the undetected error rate when the long code length is much lower than short code length. As a matter of fact, the reliability of the early stopped method is the major concern for the flash controller rather than comparing the performance. If the detection failure has occurred from the BCH decoder, the credibility of hard decoding would collapse. To address this issue, this paper focuses on the practical consideration to investigate the malfunction probability in this sense. To evaluating the credibility of the proposed method, we have given a complete test sample based on an FPGA board from the Altera family Statix II which operates at a clock rate of 110Mhz and using BCH code length of 16384 that is suitable for USB firmware testing. The system throughput is set to 480Mbps based on the USB 2.0 standard. The whole test sample quantity has a great amount of 5.9793×10^{35}. Each test sample contains the data package of 3 BCH code blocks and code length is 16383 using $GF(2^{14})$ BCH code t=72. This result means that we never encountered any decoding failure during the time using a storage device based on the proposed design. Finally, this technique has been applying to commercial USB device since 2012 and the USB controller name is BR825CA illustrated in Fig. 3.

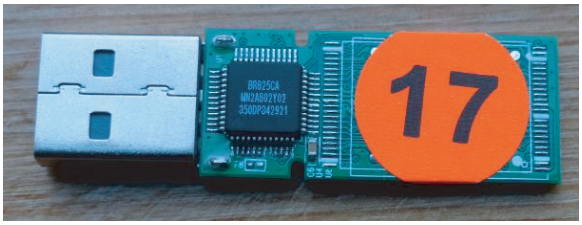

Fig. 3: A photo of the commercial product applying proposed method

IV. Conclusion

In this paper, we have provided a practical solution for early termination checking while decoding BCH code. Each version of the proposed checking strategy observes any the discrepancies during each BM iteration, and provides termination signal to reduce the number of redundant iterations when using the BM algorithm. The proposed method can also be used to shorten the latency of the BCH decoding. We have also provided numerical results that estimate the probability of undetected errors when implementing the proposed approach. The analysis illustrates that the probability of undetected errors is lower than 5.24051×10^{-26} for $GF(2^{10})$ BCH code t=17. The FPGA testing on a USB device using $GF(2^{14})$ BCH code t=72 has been implemented to justify the reliability of early termination checking strategy and the amount of testing samples is accumulated up to 5.9793×10^{35}. This approach is shown to provide a solution for a practical design.

References

[1] S. Aritome, *NAND Flash Memory Technologies*. IEEE Press Series on Microelectronic Systems: Wiley, 2015.

[2] Y. Nishi, *Advances in Non-volatile Memory and Storage Technology*. Electronic and Optical Materials: Woodhead Publishing, 2014.

[3] X. W. Wei Yan and X. Yu, "Design and implementation of an efficient flash-based ssd architecture," *Information Science and Technology (ICIST) 2014 4th IEEE International Conference on*, pp. 79–83, 2014.

[4] L. C. Yu Liu and X. Wang, "Commands scheduling optimized flash controller for high bandwidth ssd application," *Solid-State and Integrated Circuit Technology (ICSICT) 2012 IEEE 11th International Conference on*.

[5] S. C. A. K. M. W. L. Rota, M. Caselle, "A pcie dma architecture for multi-gigabyte per second data transmission," *Nuclear Science IEEE Transactions on*, vol. 62, pp. 972–976,, 2015.

[6] M.-G. Kim and J. H. Lee, "Undetected error probabilities of binary primitive bch codes for both error correction and detection," *IEEE TRANSACTIONS ON COMMUNICATIONS*, vol. 44, no. 5, pp. 575–580, May 1996.

[7] E. R. Berklekamp, *Algebraic Coding Theory*. New York, NY: McGraw-Hill, 1968.

[8] a. C.-C. L. Chih-Wei Liu, "A view of gaussian elimination applied to early-stopped berlekamp-massey algorithm," *IEEE TRANSACTIONS ON COMMUNICATIONS*, vol. 55, no. 6, pp. 1131–1143, Jun. 2007.

[9] C. L. CHEN, "High-speed decoding of bch codes," *IEEE TRANSACTIONS ON INFORMATION THEORY*, vol. 27, no. 2, pp. 254–256, 1981.

[10] D. V. Sanvate and R. D. Morrison, "Decoder malfunction in bch decoders," *IEEE TRANSACTIONS ON INFORMATION THEORY*, vol. 36, no. 4, pp. 884–889, Jul. 1990.

[11] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications (2nd edition)*. NJ: Prentice Hall, 2004.

[12] W. Peterson and E. Weldon, *Error-Correcting Codes*. Comabridge, MA: MIT Press, 1972.

[13] M. Srinivasan and D. V. Sanvate, "Malfunction in the peterson-gorenstein-zierler decoder," *IEEE TRANSACTIONS ON INFORMATION THEORY*, vol. 40, no. 5, pp. 1649–1653, 1994.