AN ADAPTIVE ECC SCHEME FOR DYNAMIC PROTECTION OF NAND FLASH MEMORIES

Liu Yuan, Huaida Liu, Pingui Jia, Yiping Yang

Department of Institute of Automation, Chinese Academy Sciences, Beijing, 100190, China.

ABSTRACT

Error correcting code (ECC) is an essential method in protection of NAND Flash memories. Complexity of it is increasing rapidly with the increment of error correction capability. Traditionally, the software implementation of ECC which has less cost and high flexibility is nearly ignored due to its inefficiency. This situation can be changed by design of faster software-based ECC scheme. We have found that the reliability is constantly changing with Program/Erase cycles and retention time during the lifetime of memories. However, regular ECC methods fail to take into account the variable kinds of reliability and result in a great waste of computations. In order to achieve better software implementation, an adaptive ECC scheme is proposed to provide various amount of protection. By assigning appropriate ECC, an adaptive ECC with Hamming and more efficient BCH codes is designed to provide corresponding protection. Proposed ECC scheme is suitable for diverse devices and applications. The coding time can be obviously decreased, making it possible to replace hardware methods with software methods.

Index Terms— Error correcting code, flash memories, adaptive protection, software, BCH

1. INTRODUCTION

NAND Flash memories are widely used for mass storage because of their low power, great storage and low cost[1]. However, they also suffer from program disturb errors and data retention errors[2]. In order to enhance reliability, error correcting code(ECC) is widely used in fault tolerant design. Hamming which is the most efficient 1-bit ECC has already been used in SLC products[3]. Since the reliability of flash memories is decreased with the growth of storage density, ECCs with higher error correction capability must be found. Bose-Chaudhuri-Hocquenghem(BCH) is one of the most popular ECCs for flash memories, which has less check bits and can adapt to random and burst errors[4].

A large amount of researches have been done to improve the efficiency of ECC, but most of the attempts are focus on hardware implementation[2, 4]. People seem to forget the software implementation methods because of their inefficiency. However, the hardware methods need to design additional hardware circuit, and have longer design-time and higher cost. At present, a serial BCH[5], a parallel BCH[6] and a fast BCH based on efficient root finding method[7] have already been introduced. Although the coding time is still long and the data throughput can only satisfy several real-time applications, it tells us the software methods can still do the error correcting job. If the efficiency of software methods can be further increased, it will be possible to replace some hardware methods.

Meanwhile we found that the complexity of ECC was rapidly increasing with the growth of error correction capability. The implementation of BCH needs a large amount of computations because of the increasing raw bit error rate(RBER). However, during the life time of memories, the RBER is changing. Research shows the errors in memories have Program/Erase(P/E) cycle dependence and retention time dependence[8]. The reliability will continuously decrease with the increment of P/E cycle count and retention time[8]. So the error-correcting requirement is dynamic and can be influenced by the effect factors. Regular ECC methods[2, 4, 5, 6, 7] are designed based on the worst reliability cases and have a great waste of computations. A reconfigurable ECC for protection of SRAM is provided in [9] to solve this problem, but it can only use hardware method and doesn't match the structure of NAND Flash. So an adaptive ECC scheme is presented here in order to take full use of the ECC and improve the software efficiency.

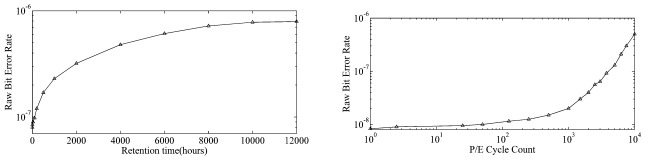
Three contributions are made in this paper: A ECC selection method based on P/E cycle count and retention time is provided to select suitable ECC. A more efficient BCH decoding method is introduced to reduce decoding time. And a structure of adaptive codec based on Hamming and BCH is designed to provide various amount of protection. When data need protection, we first determine the ECC selection according to the running environment, then the corresponding level of ECC is invoked to provide protection. No need additional hardware cost, the error correcting time running by ADSP-TS201 can be reduced to several microseconds, and the performance can be greatly improved to satisfy most real-time applications.

The rest of paper is organized as follows. Section 2 presents how to select ECC for memories. Section 3 describes the structure of proposed adaptive ECC encoder and decoder, and provides a more efficient BCH decoding method. Section 4 presents the DSP implementation of proposed ECC scheme. Section 5 concludes the paper.

2. ECC SELECTION METHOD

In this section, a method generated by considering of the effect factors of reliability is given to determine appropriate ECC. The factors such as P/E cycle count and retention time have the long-term accumulation influence on reliability[1]. The effects of these two factors are additive, the increasing of each one can result in a larger error rate[8]. We introduce them separately.

The tests in [1] are used to show the relationship between the RBER and the retention time. Fig.1a uses the test results to show the RBER versus retention time of 0 to 12000 hours under the situation of 1K P/E cycles. Note that a increasing RBER is obtained with the growth of retention time. When the retention time is increased from 1 day to 1 year, the RBER can be regarded as 8.8×10^{-8} to 6.9×10^{-7}. Then we can assume that the RBER in the retention time of 1.5 years can be about 10 times higher than that in 1-day



(a) Retention time (b) P/E cycle count

Fig. 1. RBER versus retention time and P/E cycle count for MLC devices.

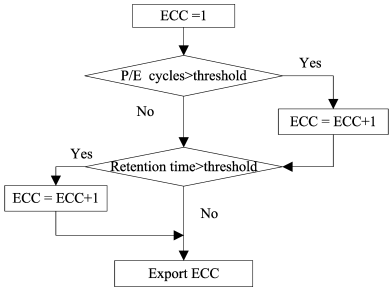


Fig. 2. Major steps in the determination of ECC.

retention time.

Except for the retention time, all types of errors are also closely connected with P/E cycle count [8]. P/E cycle count is the key factor in determining the reliability of memories. Tests in [1] show the increasing tendency of RBER with P/E cycle count. The test results of P/E cycle induced RBER when the retention time is 1 day is simulated as in Fig.1b. We can see that when the P/E cycle count is over 1K, the increasing tend becomes obviously. And when the P/E cycle count is between 1K and 10K, the RBER can be regarded as ranging from about 2×10^{-8} to 5×10^{-7}. Finally, the RBER will reach to a stable value when the P/E cycle count reaches to the maximum number.

By considering the effects of retention time and P/E cycle count, the following steps as shown in Fig.2 are done to select ECC. A threshold method is used to divide the levels of retention time and P/E cycle count. The maximum error correction capability is designed to be 4 which can satisfy most of reliability requirements [8]. The related ECC levels are divided as three: 1, 2 and 3. The level 1 uses Hamming as 1-bit ECC, the level 2 uses BCH(4122,4096,5) as 2-bit ECC and the level 3 uses BCH(4148,4096,9) as 4-bit ECC. During the lifetime of memories, the initial ECC level is 1, afterwards if the prospective retention time is larger than the threshold, the ECC level adds 1. If the P/E cycle count is also larger than the threshold, the ECC level adds another 1. After the two comparisons, the level of ECC can be selected. When data need protection, this ECC selection is output to the adaptive ECC encoder.

3. ADAPTIVE ECC ENCODER AND DECODER

The adaptive ECC encoder and decoder are the key modules in proposed ECC scheme. An adaptive 4-bit ECC codec with Hamming, BCH(4122,4096,5) and BCH(4148,4096,9) is presented to realize the maximum 4-bit error-correcting capability. The structure of adaptive ECC encoder and decoder is introduced separately.

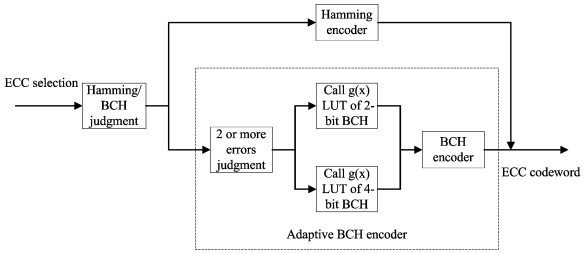


Fig. 3. Major steps in the adaptive ECC encoder.

3.1. Adaptive ECC Encoder

The structure of adaptive ECC encoder is shown in Fig.3. When data need protection, the following steps will be done:

1. The adaptive ECC encoder receives the ECC selection as mentioned in section 2, and determines which level of ECC should be chosen;

2. If the ECC level is 1, Hamming encoder[3] is applied, then turn to step 5 directly. Otherwise adaptive BCH encoder is invoked;

3. Since the procedure of BCH encoder can be written as[6]:

$$C(x) = x^{n-k}m(x) + [x^{n-k}m(x)]mod(g(x)) \quad (1)$$

Where C(x) is the generated BCH codeword, m(x) represents the information bits, n is the length of codeword, k is the length of information bits, mod is the modulus operation, g(x) is the generator polynomial. The g(x) look-up table(LUT) is used to implement BCH encoder in parallel[6]. We find that the only difference between 2-bit BCH and 4-bit BCH is g(x) LUT. So in step 3, the corresponding g(x) LUT of 2-bit or 4-bit BCH is called to provide various divisors;

4. After the injection of g(x) LUT, the BCH encoder introduced in equation (1) is invoked to calculate the ECC check bits;

5. Finally the generated ECC codeword is stored into memories.

3.2. Adaptive ECC Decoder

ECC decoder is the most complexity step in proposed ECC scheme, the improvement of it has a huge influence on efficiency. The adaptive ECC decoder takes advantage of Hamming decoder and adaptive BCH decoder to provide dynamic protection. Hamming decoder is the same with [3]. BCH decoder presented here has focused on the improvement of software performance running by DSP. During BCH decoding, the number of elements in syndrome is t, which is equal to the error correction capability. A 4-bit simplified Peterson-Gorenstein-Zierler(PGZ) method is applied to calculate the error location polynomial. The same with [7], a fast root finding method is applied to find roots of the error location polynomial, but proposed method uses Zinoviev method[10] directly, rather than BTA method. Fig.4 shows the structure of adaptive ECC decoder, which includes the BCH decoding process.

1. The number of ECC check bits is used to determine which ECC is selected. If the number of them is 24, the Hamming decoder in [3] is selected, otherwise adaptive BCH decoder is selected.

2. If the number of check bits is 26, the 2-bit BCH is selected. otherwise the 4-bit BCH is selected. In order to recognize the existence of errors as early as possible, corresponding 2-bit or 4-bit BCH encoder is invoked to encode received vector r(x) again after the ECC judgment. The new ECC check bits will be generated as ECC_{new}. Then the following steps are the same.

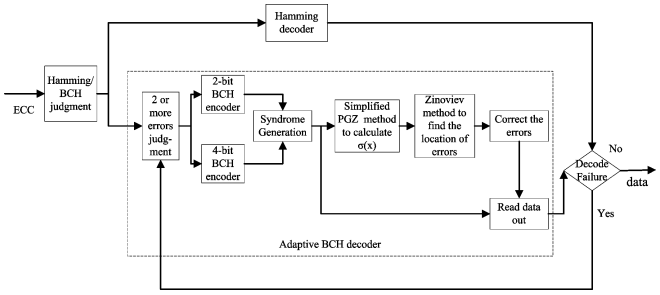


Fig. 4. Major steps in the adaptive ECC decoder.

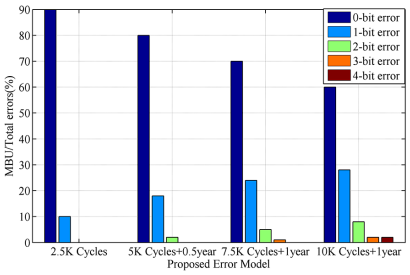


Fig. 5. Percentage of MBU errors in 4 proposed error models.

3. In step 3, XOR operation between ECC_{new} and the original ECC check bits ECC_{old} is done. If all bits in the XOR result $syn(x)$ are zero, there are no errors here. Finish the decoding and read the data out directly. Otherwise, calculate syndrome by $S_i = syn(\alpha^{2i-1})$, where $\alpha^{2i-1} \in GF(2^m)$ and $i = 1, 2 \ldots, t$. Syndrome can be obtained as $S(x) = \{S_1, S_2, \ldots, S_t\}$, and the number of elements in syndrome is $t = 4$ here.

4. After syndrome generation, the error location polynomial $\sigma(x)$ is calculated by the 4-bit simplified PGZ method, which is more suitable for DSP implementation than Berlekamp-Massey(BM) algorithm [6]. The error location polynomial can be represented as $\sigma(x) = \sigma_0 + \sigma_1 x + \ldots + \sigma_t x^t$. There are three situations:

If the syndrome satisfies: $S_2 = S_1^3$, there is 1 error in $r(x)$. The error location polynomial $\sigma(x) = 1 + S_1 x$;

Otherwise, if $S_2 S_1^3 = 0$, there are 2 errors in $r(x)$. The error location polynomial is $\sigma(x) = 1 + S_1 x + \frac{S_2 + S_1^3}{S_1} x^2$;

If the above conditions are all not satisfied, there may be 3 or 4 errors. Calculate the coefficients of error location polynomial directly by:$\sigma_0 = 1$, $\sigma_1 = S_1$, $\sigma_2 = \frac{S_1(S_4+S_1^7)+S_2(S_3+S_1^5)}{S_2(S_2+S_1^3)+S_1(S_3+S_1^5)}$, $\sigma_3 = (S_2 + S_1^3) + S_1\sigma_2$, $\sigma_4 = \frac{(S_1^2 S_2+S_3)+(S_2+S_1^3)\sigma_2}{S_1}$. In this case if $\sigma_4 = 0$, there are 3 errors in r(x), otherwise there are 4 errors in r(x).

5. Since the roots of $\sigma(x)$ are the reciprocal of the real error locations. We have to find them rapidly by the Zinoviev method[10]. Three situations are taken into account separately.

Situation 1: the case $\sigma(x) = 1 + S_1 x$, the only root of $\sigma(x)$ is: $x = S_1^{-1}$.

Situation 2: the case of 2 errors, the β_i LUT with $m \times 1$ elements is called to find the roots. The m elements satisfy that: $\beta_i^2 + \beta_i = \alpha^i + Tr(\alpha^i)\alpha^k$, and $Tr(\alpha^k) = \alpha^k + \alpha^{2k} + \alpha^{2^2 k} \ldots + \alpha^{2^{m-1} k} = 1$. Let $u = \frac{\sigma_0 \sigma_2}{\sigma_1^2} = \sum_{i=0}^{m-1} u_i \alpha^i$, where $u_i \in GF(2)$ and $\alpha^i \in GF(2^m)$. The two roots of error location polynomial can be obtained by: $x_1 = \frac{\sigma_2}{\sigma_1} \sum_{i=0}^{m-1} u_i \beta_i$ and $x_2 = x_1 + 1$.

Situation 3: the case there are 3 or 4 errors. We first change the error location polynomial $\sigma(x)$ to the equation as: $y(z) = z^4 + az^2 + bz + c$. Zinoviev[10] provides a fast method to find the roots of $y(z)$. 1) Let $L(z) = z^4 + az^2 + bz$, and $L(\alpha^i) = \sum_{j=0}^{m-1} l_{ij} \alpha^j$. Where $i = 0, 1, \ldots, m-1$, $\alpha^j \in GF(2^m)$, $l_{ij} \in GF(2)$. 2) Let $c = \sum_{j=0}^{m-1} c_j \alpha^j$, where $c_j \in GF(2)$. 3) Let $z = \sum_{j=0}^{m-1} z_j \alpha^j$, where $z_j \in GF(2)$. 4) The four roots Z_i of $y(z)$ can be calculated by the linear equation:$\sum_{i=0}^{m-1} l_{i,j} z_i = c_j$. The rank of it is 4, so we can obtain the four roots by $Z_i = \sum_{j=0}^{m-1} z_j \alpha^j$, $i = 1, 2, 3, 4$. 5) Finally the roots of error location polynomial $\sigma(x)$ can be obtained by the inverse transform of $y(z)$.

6. After the upper steps, the error locations can be found rapidly. If errors are not corrected successfully, the BCH encoder with higher error correction capability is invoked to protect the data again. Otherwise the data can be read out to the data buffer.

The architecture of proposed ECC codec method has lower complexity, which is more suitable for software method, especially the DSP implementation. Through this adaptive ECC scheme, a high speed ECC encoding and decoding can be obtained.

4. SIMULATION AND COMPARISON

In order to show the feasibility of proposed ECC scheme, several error models are introduced to simulate typical stages of running conditions. We introduce four error models during the 10K P/E cycles to obtain average performance of proposed ECC scheme. Based on the simulation of uncorrectable bit error rate(UBER), we can see a larger possibility of Multiple Bit Upset(MBU) errors with the increment of RBER in the memory block [2]. The following four error models are assumed to simulate different running environment.

During the lifetime of 10K P/E cycles, the threshold of P/E cycle count is set to be 5K and the threshold of retention time is 0.5 year, the average coding time and UBER are obtained by the injection of error models as shown in Fig.5. At the initial phase of P/E cycle count less than 2500 and retention time less than 1 day, model 1 is assumed as 10% 1-bit errors to represent the 2500 P/E operations, the P/E cycle count and retention time are less than the thresholds, the Hamming code is invoked to protect this stage. At the next 2500 P/E cycles, the retention time increases to 0.5 year which is above the threshold, model 2 is presented as 18% 1-bit errors and 2% 2-bit errors. The BCH(4122,4096,5) is selected to protect memories. When the P/E cycle count ranges from 5K to 7.5K and the retention time increases to 1 year, this relatively low reliability situation is represented by model 3. 70% 0 errors, 24% 1-bit errors, 5% 2-bit errors and 1% 3-bit errors will happen during the P/E operations. The BCH(4148,4096,9) is invoked to protect this stage. The final phase is presented by model 4. The P/E cycle count is larger than 7.5K and the retention time is 1 year. The BCH(4148,4096,9) is invoked to correct a combination of 60% 0-bit errors, 28% 1-bit errors, 8% 2-bit errors, 2% 3-bit errors and 2% 4-bit errors.

In order to show the better performance of proposed ECC scheme, two comparisons will be done. Firstly, the decoding time of BCH mentioned in section 3.2 is compared with other software implementation methods to show its efficiency. Next the comparison of proposed adaptive ECC and regular 4-bit BCH will be provided. The structure of proposed adaptive ECC can be run by various devices, such as DSP, ARM and FPGA. Since there are many judgments here, it's particularly suited for DSP implementation. The

Table 1. Comparison of the Different Implementations of BCH Decoding.

BCH decoder	2-bit(μs)	4-bit(μs)
Proposed BCH	10.01	21.345
Serial BCH[5]	609.1	1268.6
Parallel BCH[6]	134.9	287.4
Fast BCH[7]	10.69	23.924

Table 2. Comparison of Proposed ECC Scheme and 4-bit BCH.

	Encode time(μs)	Decode time(μs)	RBER (bit^{-1})	UBER (bit^{-1})
Proposed	6.90	7.99	2.1×10^{-9}	7.7×10^{-18}
4-bit BCH	9.54	10.93	2.1×10^{-9}	8.9×10^{-19}

embedded system we used is based on ADSP-TS201, the NAND Flash is connected with the DSP through FPGA. The reading and programming operations are controlled by the ADSP-TS201. All the architectures of proposed ECC system including the ECC selection and adaptive codec will be software-based implemented by ADSP-TS201 and Visual DSP++ 5.0 compiler.

We first compare the performance of proposed BCH decoding method with related software decoding methods[5, 6, 7] as shown in Table 1. Note that proposed 2-bit BCH decoding time is $10.01\mu s$, the speedup of proposed 2-bit BCH is 13.5 times larger than parallel BCH method and 1.1 times larger than fast BCH. And proposed 4-bit BCH decoding time can be reduced to $21.345\mu s$. They all show a decrease of running time. The throughput of regular 4-bit BCH by proposed BCH decoding method can reach $191.9Mbps$, which is already an acceptable speed of data transmission.

The adaptive ECC scheme can achieve another better result. The comparison between proposed ECC scheme and regular 4-bit BCH is shown in Table 2. The average efficiency is obviously improved during the lifetime of 10K P/E cycles. The encoding time is reduced from $9.54\mu s$ to $6.90\mu s$, and the decoding time is reduced from $10.93\mu s$ to $7.99\mu s$. Since the error correction capability is not waste in proposed ECC scheme, the average UBER is about 10x higher than 4-bit BCH, but it will be sure to stay below the safe UBER 10^{-15} to satisfy the reliability requirements. The decoding throughput can reach $512.6Mbps$. Although the hardware throughput can reach to over 1Gbps [4] which is about 2 times larger than proposed scheme, proposed scheme can also achieve the high speed data transmission, e.g., the MP3 needed NAND Flash data throughput is 320Kbps and the the application of H.264 baseline video is 14Mbps[6].

From the above comparisons, we can draw a conclusion that the proposed ECC scheme can achieve better results than regular ECC methods under most of situations. Through the proposed ECC scheme, we can avoid the waste of error-correcting capability and reduce the coding time to ensure the throughput to fit real-time requirements. It's possible to replace the place of several hardware methods to obtain lower cost and higher flexibility.

5. CONCLUSION

In this paper, an adaptive ECC scheme is proposed to provide dynamic protection of flash memories. A ECC selection method and an adaptive ECC with 4-bit error correction capability are designed to provide variable protection. The coding time is obviously optimized by assigning appropriate ECC and using efficient ECC implementation method. The architecture of proposed ECC scheme has high flexibility, low cost, low complexity and can adapt to various devices. The ADSP-TS201 based average encoding time can be decreased by 28%, the decoding time is reduced by 27%. The data throughput of proposed ECC scheme can reach $512.6Mbps$ under the situation of average UBER below 10^{-15}, which is only 2 times less than hardware throughput. So proposed ECC scheme can replace the place of several hardware methods and obtain an acceptable data transmission speed. The future investigation will include the further study of reliability factors and find more reasonable ECC selection method.

6. REFERENCES

[1] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. R. Nevill, "Bit error rate in NAND flash memories," in *Proc. IEEE Int. Rel. Phys. Symp.* IEEE, 2008, pp. 9–19.

[2] C. Yang, D. Muckatira, A. Kulkarni, and C. Chakrabarti, "Data storage time sensitive ECC schemes for MLC NAND flash memories," in *Proc. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*. IEEE, May 2013, vol. 5, pp. 2513 – 2517.

[3] Samsung, "NAND Flash ECC algorithm (error checking & correction)," `www.elnec.com/sw/samsung_ecc_algorithm_for_256b.pdf`, 2004.

[4] Y. Chen and K.K. Parhi, "Area efficient parallel decoder architecture for long BCH codes," in *Proc. 2004 IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP'04)*. IEEE, 2004, vol. 5, pp. 73–76.

[5] Micron, "Enabling software BCH ECC on a linux platform introduction," `http://www.micron.com/~/media/Documents/Products/Technical%20Note/NAND%20Flash/tn2971_software_bch_ecc_on_linux.pdf`, 2004.

[6] J. Cho and W. Sung, "Efficient software-based encoding and decoding of BCH codes," *IEEE Trans. Comput.*, vol. 58, no. 7, pp. 878–889, Jul. 2009.

[7] B. Biswas and Vi. Herbert, "Efficient root finding of polynomials over fields of characteristic 2," `http://hal.archives-ouvertes.fr/docs/00/62/69/97/PDF/tbz.pdf`, Sept. 2011.

[8] Y. Cai, E.F. Haratsch, O. Mutlu, and K. Mai, "Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis," in *Proc. of the Design, Automation & Test in Europe Conference & Exhibition (DATE) 2012*. IEEE, Aug. 2012, pp. 521–526.

[9] S. Paul, F. Cai, X. Zhang, and S. Bhunia, "Reliability-driven ECC allocation for multiple bit error resilience in processor cache," *IEEE Trans. Comput.*, vol. 60, no. 1, pp. 20–34, Jan. 2011.

[10] V.A. Zinoviev, "On the solution of equations of degree ≤ 10 over finite fields GF(2^m)," *INRIA, Rapport de Recherche n^o 2829, Institute National de Recherche en Informatique et en Automatique*, 1996.