Synthesis of BCH Codes for Enhancing Data Integrity in Flash Memories

Rajesh Shetty K , Sripati U, Prashantha Kumar H
Electronics & Communication Department
National Institute of Technology Karnataka, Surathkal
Mangalore, India.

B.Shankarananda
Mangalore Institute of Technology & Engineering
Mijar, Moodabidri
India.

Abstract— **Flash memories have found extensive application for use in portable storage devices. They have been used for code storage as well as data storage. The storage density associated with these devices has increased tremendously in the past few years. This has necessitated very dense packing of data bits on the device. This gives rise to increased Raw Bit Error Rate (RBER) as a result of Inter Symbol Interference (ISI) between bits stored in adjacent cells. This necessitates the use of powerful error control codes to guarantee information integrity. With the increase in density of data storage, the raw bit error rate (RBER) associated with the storage device increases. Error Control Coding (ECC) can be used to reduce the RBER to acceptable values so that these devices can be employed to store information in applications where data corruption is unacceptable. In this paper, we describe the synthesis of BCH codes based on memory models proposed by the semiconductor industry. These codes have better error correcting capability than the codes used in current practice.**

Keywords - BCH code, Flash memory, Bit Error Rate (BER), Raw Bit Error Rate (RBER), Uncorrectable Bit Error Rate (UBER), Congugacy constraints

I. INTRODUCTION

Data storage devices have become ubiquitous in present day information driven society. It is very essential that storage devices exhibit very high levels of data integrity. Therefore, data integrity is a fundamental aspect of storage, security and reliability. NAND and NOR Flash memories [4], [6] are widely used for data storage because of their compactness and low power consumption. Data stored in nonvolatile memory is usually critical to proper system operation, and corruption of data can lead to system failure. Hence data corruption is a major concern in applications that rely on nonvolatile memory for long-term data storage. Many techniques have been employed to improve the reliability of these devices. These techniques can be divided into two categories. In the first approach, improvements are carried out in the fabrication process to reduce the Raw Bit Error Rate (RBER). The second option is to use Error Correction Techniques to improve the RBER level to levels deemed acceptable to most users [7].

ECC is a cost effective method to enhance the integrity of data storage systems. The fraction of bits that contain incorrect data before applying ECC is called Raw Bit Error Rate (RBER). Very stringent values of application BER, which would ordinarily require complex and expensive fabrication techniques as well as expensive materials, can be met very easily by employing ECC. Storage devices characterized by high RBER values can be made to yield application BERs as small as desired by the use of suitable ECC techniques. The fraction of erroneous bits that remain uncorrected after applying ECC constitute the uncorrectable bit error rate (UBER). UBER is a useful reliability metric for data storage devices such as hard disk drives and flash memories. UBER is used to specify the data corruption rate in the information given to the user after correction by ECC algorithms. ECC algorithms can also correct errors that may manifest at any later stage during the life of the device. Hence use of ECC techniques has been widely accepted by the semiconductor manufacturing industry to enhance the RBER to levels demanded by applications.

ECC schemes supplement user data with redundant information (parity bits) which store enough extra information for the data to be reconstructed if one or more bits are corrupted. Many ECC algorithms have been employed to correct errors in storage systems. We have synthesized a number of codes for use in Flash memories with error correcting capabilities exceeding the state of art as specified in industry documentation [4]. These codes and their performance have been presented in the paper

II. MOTIVATION FOR THE USE OF ECC IN STORAGE SYSTEMS

As recording densities increase, a very large number of bits have to be packaged into a very small physical area. Consequently, the physical space available to accommodate a bit has become smaller and smaller over the years. This results in Inter Symbol Interference (ISI) in the sense that the detection of an information bit is influenced by bits that are present in the recording medium in the immediate vicinity. This problem becomes more acute with increase in recording density. The use of powerful error control algorithms can protect the integrity of user information against errors caused by ageing, wear out due to repeated read and write operations and manufacturing defects. In this work we are proposing a solution to the problem of ensuring data integrity based on BCH codes for use in flash memories. As compared to the current standard [1], [4], where 6 bits in error can be corrected over a span of 4096 information bits (512 bytes), we propose codes that can correct up to 19 bits in error over a span of 512 x 2 = 1024 bytes (8192 bits). In flash memories, the storage space is organized into blocks, pages and sectors. The smallest unit is a sector. There are two memory models used in current practice. In the first model (Memory model-I), each sector has 512 bytes reserved for storing information and 16 bytes reserved for storing parity information [1]. In the second model (Memory model-II), each sector has 512 bytes reserved for storing information and 32 bytes reserved for storing parity check (redundant) information [1]. Figure 1 shows the organization of sectors, pages and blocks in 2GB flash memory.

Sector size	512 bytes
Sector/page	8
Pages/block	64
Page size	4 KB
Block size	256 KB
Blocks/die	4096
Dies/chip	2
Total capacity	2 GB

Figure 1. Memory organization for a 2GB flash memory.

The problem of designing an error control scheme by considering 512 bytes (one sector) as an information block and 16 bytes (128 bits) to store redundant information has been addressed by us in [12]. In this paper, we have described the synthesis of BCH codes with error correcting capability of t=8 and t=9 bits per 512 bytes (one sector) which exceeds the state of art (6 bits per 512 bytes) [1], [4]. We have sought to present more powerful and versatile codes in the present paper. The codes proposed by us in this paper can be divided into two categories:

(i) Codes base on Memory model-I where two sectors have been combined to yield a single information block of size 1024 bytes and redundancy allocation of 16 x 2=32 bytes.

(ii) Codes based on Memory Model-II in which each information sector comprises of 512 bytes and the allocation for holding overhead information is 32 bytes.

The choice of the ECC to be used in a particular application depends upon many factors such as, organization of information and overhead data in the storage devices, types of errors, computational complexity permitted by the architecture of the storage devices, latency tolerable in a given application etc. We have synthesized BCH codes where the smallest information block size is 1024 bytes in the first category and 512 bytes in the second category. We have desisted from synthesizing codes with smaller information blocks (128 bytes or 256 bytes) because this results in inefficient use of overhead resources.

III. STATE OF ART

The application of BCH codes in Flash memories for error control has been discussed in [1], [2], [3], [4], [5]. In [4], [5] and [6], the performance of various ECC schemes used for enhancing the reliability of Flash memories has been documented. As per these references, the best performing BCH codes known are able to correct as many as t = 6 bits in error (t represents the error correcting capability of an error control code) over a span of 4096 bits (1 sector) (refer Figure.1). In this paper, we have synthesized various codes meeting the requirement of the memory architecture that exceed the state of art. In (4), (6) and (8), we specify the generator polynomials of BCH code that can correct t = 17, t = 18 and t = 19 errors over a span of 8192 information bits. The motivation for synthesizing these codes is that the performance of an ECC generally improves with increase in length of the code [11]. In (10), we specify the generator polynomial of a BCH code that can correct t = 19 errors over a span of 8192 bits. However, this code requires an overhead of 266 bits for storing redundant information produced by the ECC as against the allocation of 256 bits provided by the memory model. Hence, this code is not suitable for use in flash memories with the present architecture. The performance of these codes is quantified by computing values of the probability of decoding error as specified in (2). We have derived performance plots for these codes which have been documented in Figure 3. Figure 4 shows the performance plot for Memory model-II with t=18 and t=19. These plots demonstrate that the error control codes synthesized by us offer substantial improvement in the level of reliability (quantified as the probability of decoding error) over codes constituting the current state of art. In Figure 2 [4], the performance of some codes used in the current practice has been documented. In Table III and Table IV, we have presented data that quantifies the improvement resulting from the use of these codes.

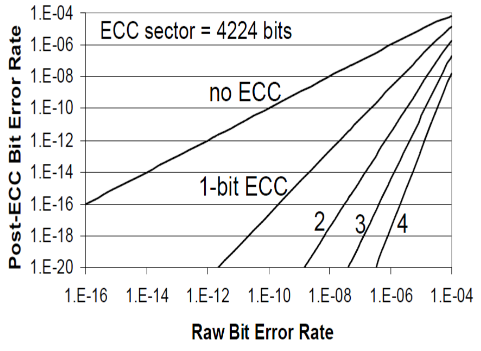

Figure 2. Effect of ECC [4].

IV. CODE SYNTHESIS – BCH CODES

A. *Memory model-I (Two sectors as one information block)*

BCH codes form a large class of powerful random error correcting codes. By employing the BCH bound, it is possible to synthesize codes which possess distance properties as required by the given application. Encoding and decoding of these codes can be implemented without excessive circuit complexity. It is for these reasons that BCH codes have found widespread use in many applications. The code is capable of correcting any combination of t or fewer errors in a block of length n ($n|q^m$-1) digits. This is called a t error correcting BCH code. BCH codes are cyclic codes and hence may be specified by a generator polynomial. The generator polynomial of this code is specified in terms of its roots from the Galois field F_{q^m} where m is the smallest positive integer such that $n|q^m$-1. Design of BCH codes is discussed in [8], [9], [10], and [11].

To design a BCH code of length n over F_q , we choose the smallest m such that $n|q^m$-1, where q is a power of prime. In practical applications usually q = 2. Hence we choose the smallest m such that $n|$ 2^m-1. BCH codes, take advantage of a useful result that ensures a minimum "design distance" given a particular constraint on the generator polynomial. This result is known as the *BCH bound* [8], [9], [10], and [11]. An (n, k) block code is characterized by the value of n and k. The process used for synthesis of codes is as follows:

i) Given the values of (n, k), n-k represents the number of parity symbols. ($n > k$). Choose the smallest value of m such that $n|2^m$-1. (i.e. if k = 4096 = 2^{12}, then shortest primitive length BCH code with n > 4096 would have a length n = 2^{13}-1=8191).

ii) Select the appropriate extension field of F_{2^m} in which the primitive nth root of unity can be found. (in this case $F_{2^{13}}$)

iii) Invoking the BCH bound, determine the elements.

$\{\alpha^b, \alpha^{b+1}, \alpha^{b+2}, ..., \alpha^{b+\delta-2}\}$ of F_{2^m} which serve as the required roots of the generator polynomial (b≥1 and δ is the design distance). The generator polynomial must also possess certain extraneous roots imposed by conjugacy constraints [8], [9], [10], [11].

iv) The extraneous roots are determined by referring to the conjugacy class of each required root. After identifying the conjugacy classes, we compute the associated minimal polynomials. The minimal polynomial of α^b is specified as $M_b(x)$, that of α^{b+1} is specified as $M_{b+1}(x)$ ...... the minimal polynomial of $\alpha^{b+\delta-2}$ is specified as $M_{b+\delta-2}(x)$.

v) Following the requirements of the BCH bound, the generator polynomial is computed as,

$$g(x) = LCM\{M_b(x), M_{b+1}(x), ...M_{b+\delta-2}(x)\} \tag{1}$$

Taking advantage of the fact that increasing length of the code generally improves the error correcting capability, we have sought to create long code words by combining two sectors into one information block (1024 bytes = 8192 bits). We shall demonstrate the advantage gained by doing this subsequently in this section. The corresponding overhead allocation now increases to 32 bytes. Generator polynomials of BCH codes capable of correcting t=15, t=16, t=17, t=18 bits over the span of 1024 bytes have been computed. While the BCH code capable of correcting t=19 bits can be synthesized, it cannot be applied due to constraints imposed by the memory architecture. To assess the performance of these codes, in improving the RBER, we have made use of the fact that the probability of decoding error associated with a linear code that can correct t errors per codeword can be expressed as

$$P_{decoding\ error} = \sum_{k=t+1}^{n} \binom{n}{k} (RBER)^k (1 - RBER)^{n-k} \tag{2}$$

With two sectors used as one information block, the number of information bits becomes k = 8192. We start with a narrow sense primitive BCH code of length n = 2^{14}-1 =16383. A primitive (2^{14}-1)th root of unity can be found in $F_{2^{14}}$. Hence the roots of $g(x)$ have to be located in this field. Let us examine the steps required to synthesize a t=17 BCH code. In this case, t = 17, $\delta = 2t+1 = 35$, $b = 1$, $b+\delta-2 = 34$ .Hence the required roots are:

$$\{\alpha, \alpha^2, \alpha^3, .\alpha^4, \alpha^5, \alpha^6, \alpha^7, \alpha^8, \alpha^9, \alpha^{10}, \alpha^{11}, \alpha^{12}, \alpha^{13}, \alpha^{14}, \alpha^{15}, \alpha^{16}, \alpha^{17}, \alpha^{18}, \alpha^{19}, \alpha^{20}, \alpha^{21}, \alpha^{22}, \alpha^{23}, \alpha^{24}, \alpha^{25}, \alpha^{26}, \alpha^{27}, \alpha^{28}, \alpha^{29}, \alpha^{30}, \alpha^{31}, \alpha^{32}, \alpha^{33,}, \alpha^{34}\}$$

The required roots for other values of t can similarly be computed by invoking the BCH bound. The minimal polynomials of the required roots for BCH codes corresponding to t=16, 17, 18, 19 are listed in Table I. The generator polynomials are computed and tabulated in (4), (6) and (8).

Table I. List of minimal polynomials of the required roots for BCH codes with t=16, 17, 18, 19

$M_1(x) = 1 + x + x^6 + x^{10} + x^{14}$
$M_3(x) = 1 + x + x^2 + x^5 + x^8 + x^{14}$
$M_5(x) = 1 + x + x^3 + x^4 + x^6 + x^7 + x^9 + x^{10} + x^{14}$
$M_7(x) = 1 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{12} + x^{14}$
$M_9(x) = 1 + x^2 + x^3 + x^5 + x^{11} + x^{12} + x^{14}$
$M_{11}(x) = 1 + x + x^6 + x^8 + x^{14}$
$M_{13}(x) = 1 + x^5 + x^6 + x^9 + x^{10} + x^{11} + x^{12} + x^{13} + x^{14}$
$M_{15}(x) = 1 + x^2 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{14}$
$M_{17}(x) = 1 + x + x^2 + x^3 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{14}$
$M_{19}(x) = 1 + x + x^6 + x^{11} + x^{14}$
$M_{21}(x) = 1 + x^2 + x^4 + x^8 + x^{10} + x^{11} + x^{14}$
$M_{23}(x) = 1 + x^2 + x^5 + x^6 + x^9 + x^{11} + x^{14}$
$M_{25}(x) = 1 + x + x^6 + x^7 + x^{10} + x^{11} + x^{12} + x^{13} + x^{14}$
$M_{27}(x) = 1 + x + x^7 + x^8 + x^{10} + x^{13} + x^{14}$
$M_{29}(x) = 1 + x + x^2 + x^3 + x^5 + x^8 + x^{11} + x^{13} + x^{14}$
$M_{31}(x) = 1 + x^3 + x^4 + x^7 + x^9 + x^{10} + x^{11} + x^{12} + x^{14}$
$M_{33}(x) = 1 + x + x^2 + x^3 + x^4 + x^6 + x^7 + x^8 + x^{10} + x^{12} + x^{14}$
$M_{35}(x) = 1 + x + x^2 + x^4 + x^5 + x^6 + x^{11} + x^{13} + x^{14}$
$M_{37}(x) = 1 + x + x^3 + x^5 + x^6 + x^{13} + x^{14}$

(i) t =17

In this case, the generator polynomial is computed as the LCM of the minimal polynomials of the required roots. Hence,

$$g(x) = LCM\{M_1(x), M_3(x), M_5(x), M_7(x), M_9(x),\\ M_{11}(x), M_{13}(x), M_{15}(x), M_{17}(x), M_{19}(x), M_{21}(x),\\ M_{23}(x), M_{25(}(x), M_{27}(x), M_{29}(x), M_{31}(x), M_{33}(x)\} \quad (3)$$

Upon computation of the least common multiple (LCM) of these polynomials, the generator polynomial is specified as expressed in (4).

$$g(x) = 1 + x^2 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{15} + x^{20} + x^{21} +\\ x^{23} + x^{26} + x^{27} + x^{29} + x^{34} + x^{35} + x^{36} + x^{38} + x^{39} + x^{40} + x^{42} +\\ x^{43} + x^{46} + x^{47} + x^{53} + x^{57} + x^{60} + x^{62} + x^{64} + x^{65} + x^{66} + x^{67} +\\ x^{68} + x^{71} + x^{72} + x^{73} + x^{75} + x^{76} + x^{80} + x^{81} + x^{83} + x^{86} + x^{87} +\\ x^{90} + x^{91} + x^{92} + x^{95} + x^{96} + x^{97} + x^{98} + x^{99} + x^{102} + x^{106} + x^{107} +\\ x^{109} + x^{110} + x^{115} + x^{119} + x^{121} + x^{122} + x^{123} + x^{124} + x^{125} + x^{126} +\\ x^{128} + x^{129} + x^{130} + x^{133} + x^{137} + x^{138} + x^{144} + x^{145} + x^{148} + x^{149} +\\ x^{153} + x^{154} + x^{155} + x^{157} + x^{160} + x^{161} + x^{162} + x^{163} + x^{164} + x^{167} +\\ x^{169} + x^{172} + x^{174} + x^{176} + x^{179} + x^{183} + x^{184} + x^{185} + x^{188} + x^{189} +\\ x^{193} + x^{199} + x^{201} + x^{208} + x^{209} + x^{210} + x^{211} + x^{212} + x^{213} +\\ x^{214} + x^{216} + x^{217} + x^{219} + x^{221} + x^{224} + x^{226} + x^{228} + x^{229} +\\ x^{231} + x^{232} + x^{235} + x^{237} + x^{238} \quad (4)$$

Here *degree* {*g(x)*} =238 = *n* – *k, n* = *k* + *238* =8192 + 238 =8430. The primitive BCH code parameters are (16,383, 16,383 – 238) = (16,383, 16,145). In this case, *k* = 8192 and 16145 – 8192 =7953. Therefore the parameters of the shortened BCH code are (8430, 8192). Since (*n* – *k*) = 256 and degree {*g(x)*} = 238, the overhead requirement of the code can be easily met by the Memory model-I.

Since, 256 bits of overhead space is available and only 238 bits of overhead space are utilized by the *t*=17 BCH code, we have attempted to synthesize a more powerful BCH code characterized by *t*=18.

(ii) *t* = 18

$$g(x) = LCM\{M_1(x), M_3(x), M_5(x), M_7(x), M_9(x),\\ M_{11}(x), M_{13}(x), M_{15}(x), M_{17}(x), M_{19}(x), M_{21}(x),\\ M_{23}(x), M_{25(}(x), M_{27}(x), M_{29}(x), M_{31}(x), M_{33}(x), M_{35}(x)\} \quad (5)$$

$$g(x) = 1 + x + x^3 + x^6 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{13} + x^{14} + x^{16} +\\ x^{19} + x^{20} + x^{23} + x^{32} + x^{33} + x^{34} + x^{39} + x^{40} + x^{41} + x^{42} + x^{43} + x^{44} +\\ x^{49} + x^{50} + x^{53} + x^{54} + x^{58} + x^{59} + x^{60} + x^{61} + x^{62} + x^{65} + x^{67} + x^{70} +\\ x^{72} + x^{73} + x^{75} + x^{77} + x^{79} + x^{80} + x^{82} + x^{83} + x^{85} + x^{90} + x^{91} + x^{94} +\\ x^{97} + x^{98} + x^{99} + x^{102} + x^{104} + x^{105} + x^{108} + x^{109} + x^{112} + x^{116} + x^{118} +\\ x^{119} + x^{120} + x^{121} + x^{122} + x^{124} + x^{125} + x^{127} + x^{130} + x^{132} + x^{133} + x^{137} +\\ x^{138} + x^{140} + x^{141} + x^{146} + x^{148} + x^{149} + x^{150} + x^{153} + x^{155} + x^{156} + x^{158} +\\ x^{161} + x^{163} + x^{164} + x^{165} + x^{168} + x^{169} + x^{170} + x^{172} + x^{173} + x^{177} + x^{178} +\\ x^{179} + x^{184} + x^{187} + x^{191} + x^{192} + x^{193} + x^{195} + x^{197} + x^{200} + x^{202} + x^{203} +\\ x^{208} + x^{211} + x^{214} + x^{216} + x^{217} + x^{218} + x^{219} + x^{224} + x^{226} + x^{227} +\\ x^{228} + x^{230} + x^{233} + x^{235} + x^{236} + x^{239} + x^{241} + x^{242} + x^{250} + x^{252} \quad (6)$$

Here *degree* {*g(x)*} =252 = *n* – *k, n* = *k* + *252* =8192 + 252 =8444. The primitive BCH code parameters are (16,383, 16,383 – 252) = (16,383, 16,131). In this case, *k* = 8192, and 16131 – 8192 =7939. Therefore the parameters of the shortened BCH code are (8444, 8192). Since (*n* – *k*) = 256 and degree {*g(x)*} = 252, the overhead requirement of the code can be easily met by the Memory model-I.

(iii) Finally, we consider the synthesis of a *t* = 19 BCH code for this application. In this case,

$$g(x) = LCM\{M_1(x), M_3(x), M_5(x), M_7(x), M_9(x),\\ M_{11}(x), M_{13}(x), M_{15}(x), M_{17}(x), M_{19}(x), M_{21}(x),\\ M_{23}(x), M_{25(}(x), M_{27}(x), M_{29}(x), M_{31}(x), M_{33}(x),\\ M_{35}(x), M_{37}(x)\} \quad (7)$$

$$g(x) = 1 + x^2 + x^5 + x^7 + x^8 + x^{10} + x^{13} + x^{14} + x^{16} + x^{18} + x^{19} + x^{20} +\\ x^{25} + x^{26} + x^{30} + x^{34} + x^{37} + x^{39} + x^{40} + x^{42} + x^{43} + x^{45} + x^{46} + x^{47} +\\ x^{48} + x^{49} + x^{50} + x^{54} + x^{55} + x^{57} + x^{58} + x^{60} + x^{61} + x^{63} + x^{67} + x^{70} +\\ x^{71} + x^{74} + x^{75} + x^{78} + x^{79} + x^{80} + x^{81} + x^{82} + x^{83} + x^{84} + x^{85} + x^{86} +\\ x^{90} + x^{93} + x^{94} + x^{95} + x^{98} + x^{100} + x^{101} + x^{102} + x^{104} + x^{105} + x^{106} +\\ x^{107} + x^{109} + x^{113} + x^{115} + x^{117} + x^{121} + x^{128} + x^{130} + x^{131} + x^{134} + x^{136} +\\ x^{137} + x^{139} + x^{140} + x^{144} + x^{146} + x^{147} + x^{148} + x^{149} + x^{150} + x^{151} + x^{152} +\\ x^{154} + x^{157} + x^{158} + x^{161} + x^{164} + x^{167} + x^{170} + x^{172} + x^{176} + x^{185} + x^{187} +\\ x^{188} + x^{189} + x^{190} + x^{191} + x^{192} + x^{196} + x^{198} + x^{199} + x^{201} + x^{209} + x^{210} +\\ x^{211} + x^{212} + x^{217} + x^{220} + x^{221} + x^{225} + x^{226} + x^{228} + x^{230} + x^{231} + x^{233} +\\ x^{238} + x^{241} + x^{242} + x^{244} + x^{247} + x^{251} + x^{253} + x^{254} + x^{257} + x^{258} + x^{263} +\\ x^{264} + x^{265} + x^{266} \quad (8)$$

Here *degree* {*g(x)*} =266 = *n* – *k, n* = *k* + *266* =8192 + 266 =8458. The primitive BCH code parameters are (16,383, 16,383 – 266) = (16,383, 16,117). In this case, *k* = 8192, and 16117 – 8192 =7925. Therefore the parameters of the shortened BCH code are (8458, 8192). Since (*n* – *k*) = 256 and degree {*g(x)*} = 266, a *t* = 19 BCH code cannot be accommodated in this memory model. However, if the memory allocated to store the redundant overhead bits is increased from 32 to 34 bytes then this solution can be employed.

Thus, we come to the conclusion that given the constraints of the memory architecture, it is not possible to synthesize BCH codes with error correcting capability *t* > 18 bits in this application.

In the next sub section, we have synthesized BCH codes capable of correcting *t*=18 and *t*=19 errors per sector (512 bytes). In doing this, we have adhered to the constraints imposed by the specifications of Memory model-II.

B. *Memory Model – II*

In this model, each sector has 512 bytes reserved for data and 32 bytes reserved for storing overhead information. We will use BCH codes with natural length *n* = 2^{13}-1 = 8191.

$(n-k)|_{\max} = 256$.

Choosing $b = 1,\ \delta = 2t + 1$.

For *t* = 15, $\delta = 2t + 1$ = 31.

Required roots are: $\{\alpha, \alpha^2, \alpha^3, \ldots\ldots\ldots, \alpha^{b+\delta-2} = \alpha^{30}\}$ .The minimal polynomials required for the computation of the generator polynomial of the BCH code with *t* = 18, *t* =19 and *t*

= 20 are shown in Table II. The generator polynomials of these codes are tabulated in (9), (10) and (11).

Table II: List of minimal polynomials of the required roots for BCH codes with $t = 18, 19, 20$

$$\boxed{\begin{aligned}
&M_1(x)=1+x+x^3+x^4+x^{13}\\
&M_3(x)=1+x+x^4+x^5+x^7+x^9+x^{10}+x^{13}\\
&M_5(x)=1+x+x^4+x^7+x^8+x^{11}+x^{13}\\
&M_7(x)=1+x+x^2+x^3+x^6+x^8+x^9+x^{10}+x^{13}\\
&M_9(x)=1+x^5+x^6+x^7+x^8+x^{12}+x^{13}\\
&M_{11}(x)=1+x+x^5+x^7+x^8+x^9+x^{13}\\
&M_{13}(x)=1+x^3+x^4+x^5+x^6+x^{12}+x^{13}\\
&M_{15}(x)=1+x+x^2+x^3+x^4+x^5+x^7+x^9+x^{13}\\
&M_{17}(x)=1+x+x^2+x^3+x^4+x^5+x^6+x^7+\\
&x^8+x^9+x^{10}+x^{11}+x^{13}\\
&M_{19}(x)=1+x^3+x^5+x^9+x^{11}+x^{12}+x^{13}\\
&M_{21}(x)=1+x+x^4+x^6+x^7+x^8+x^{11}+x^{12}+x^{13}\\
&M_{23}(x)=1+x+x^2+x^5+x^{11}+x^{12}+x^{13}\\
&M_{25}(x)=1+x^2+x^3+x^4+x^6+x^8+x^{10}+x^{12}+x^{13}\\
&M_{27}(x)=1+x^2+x^4+x^8+x^9+x^{12}+x^{13}\\
&M_{29}(x)=1+x^2+x^6+x^8+x^9+x^{10}+x^{11}+x^{12}+x^{13}\\
&M_{31}(x)=1+x^2+x^3+x^6+x^8+x^{10}+x^{11}+x^{12}+x^{13}\\
&M_{33}(x)=1+x+x^2+x^3+x^4+x^5+x^{10}+x^{11}+x^{13}\\
&M_{35}(x)=1+x+x^3+x^4+x^5+x^7+x^8+x^9+x^{11}+x^{12}+x^{13}\\
&M_{37}(x)=1+x^2+x^3+x^4+x^7+x^8+x^{13}\\
&M_{39}(x)=1+x+x^4+x^5+x^6+x^7+x^8+x^9+x^{10}+x^{12}+x^{13}\\
&M_{41}(x)=1+x^2+x^5+x^6+x^7+x^8+x^9+x^{11}+x^{13}\\
&M_{43}(x)=1+x^2+x^3+x^4+x^5+x^6+x^8+x^{10}+x^{11}+x^{12}+x^{13}
\end{aligned}}$$

(i) $t = 18$

$$\begin{aligned}
g(x) &= 1+x^3+x^4+x^5+x^6+x^9+x^{10}+x^{11}+x^{12}+x^{13}+x^{14}+x^{17}+\\
&x^{18}+x^{19}+x^{25}+x^{27}+x^{28}+x^{33}+x^{36}+x^{39}+x^{42}+x^{45}+x^{47}+x^{48}+\\
&x^{50}+x^{54}+x^{61}+x^{62}+x^{66}+x^{68}+x^{69}+x^{70}+x^{72}+x^{73}+x^{74}+x^{75}+\\
&x^{77}+x^{78}+x^{79}+x^{80}+x^{85}+x^{88}+x^{92}+x^{93}+x^{94}+x^{98}+x^{101}+x^{102}+\\
&x^{104}+x^{105}+x^{107}+x^{109}+x^{112}+x^{114}+x^{115}+x^{117}+x^{118}+x^{123}+x^{125}+\\
&x^{126}+x^{127}+x^{130}+x^{131}+x^{133}+x^{135}+x^{136}+x^{138}+x^{140}+x^{142}+x^{143}+\\
&x^{144}+x^{146}+x^{149}+x^{150}+x^{154}+x^{155}+x^{158}+x^{159}+x^{163}+x^{164}+x^{165}+\\
&x^{166}+x^{167}+x^{168}+x^{170}+x^{171}+x^{172}+x^{173}+x^{175}+x^{180}+x^{181}+x^{184}+\\
&x^{188}+x^{191}+x^{192}+x^{194}+x^{196}+x^{197}+x^{198}+x^{200}+x^{202}+x^{203}+x^{204}+\\
&x^{206}+x^{212}+x^{213}+x^{215}+x^{216}+x^{217}+x^{219}+x^{225}+x^{227}+x^{230}+\\
&x^{231}+x^{234}
\end{aligned} \tag{9}$$

Here *degree* $\{g(x)\}$ =234 = $n - k$, $n = k + 234$ =4096 + 234 = 4330. The primitive BCH code parameters over $F_{2^{13}}$ are (8191, 8191 – 234) = (8191, 7957). In this case, $k = 4096 \Rightarrow$ 7957 – 4096 =3861. Therefore the parameters of the shortened BCH code are (8191 - 3861, 7957 - 3861) = (4330, 4096). Since (n – k) = 256 and degree $\{g(x)\}$ = 234, the overhead requirement of the code can be easily met this memory model.

(ii) $t = 19$

Since, 256 bits of overhead space is available and only 234 bits of overhead space are utilized by the t=18 BCH code, we have attempted to synthesize a more powerful BCH code characterized by t=19. In this case,

$$\begin{aligned}
g(x) &= 1+x^2+x^6+x^9+x^{10}+x^{12}+x^{13}+x^{14}+x^{15}+x^{16}+x^{18}+x^{19}+x^{21}+x^{26}+\\
&x^{30}+x^{31}+x^{32}+x^{34}+x^{35}+x^{36}+x^{37}+x^{40}+x^{41}+x^{46}+x^{47}+x^{54}+x^{55}+x^{58}+\\
&x^{60}+x^{61}+x^{67}+x^{68}+x^{74}+x^{75}+x^{76}+x^{79}+x^{80}+x^{86}+x^{89}+x^{93}+x^{95}+x^{98}+\\
&x^{99}+x^{100}+x^{101}+x^{102}+x^{103}+x^{104}+x^{105}+x^{108}+x^{109}+x^{116}+x^{117}+x^{119}+x^{121}+\\
&x^{124}+x^{125}+x^{126}+x^{128}+x^{129}+x^{130}+x^{133}+x^{136}+x^{137}+x^{140}+x^{143}+x^{144}+x^{145}+\\
&x^{147}+x^{152}+x^{154}+x^{156}+x^{157}+x^{159}+x^{160}+x^{161}+x^{162}+x^{164}+x^{165}+x^{166}+x^{168}+\\
&x^{169}+x^{170}+x^{173}+x^{174}+x^{181}+x^{188}+x^{189}+x^{190}+x^{191}+x^{192}+x^{195}+x^{197}+x^{200}+\\
&x^{203}+x^{205}+x^{206}+x^{208}+x^{209}+x^{211}+x^{213}+x^{221}+x^{223}+x^{225}+x^{227}+x^{229}+x^{230}+\\
&x^{232}+x^{233}+x^{236}+x^{239}+x^{240}+x^{241}+x^{242}+x^{243}+x^{244}+x^{247}
\end{aligned} \tag{10}$$

Here *degree* $\{g(x)\}$ =247 = $n - k$, $n = k + 247$ =4096 + 247 = 4343. The primitive BCH code parameters over $F_{2^{13}}$ are (8191, 8191 – 247) = (8191, 7944). In this case, $k = 4096 \Rightarrow$ 7944 – 4096 =3848. Therefore the parameters of the shortened BCH code are (8191 - 3848, 7944 - 3848) = (4343, 4096). Since (n – k) = 256 and degree $\{g(x)\}$ = 247, the overhead requirement of the code can be easily met this memory model. Therefore, finally we consider the synthesis BCH code characterized by t=20.

(iii) $t = 20$

$$\begin{aligned}
g(x) &= 1+x+x^2+x^3+x^4+x^5+x^6+x^7+x^9+x^{10}+x^{11}+x^{13}+x^{17}+x^{18}+\\
&x^{19}+x^{20}+x^{21}+x^{22}+x^{23}+x^{25}+x^{26}+x^{28}+x^{29}+x^{33}+x^{35}+x^{37}+x^{40}+x^{42}+\\
&x^{44}+x^{47}+x^{48}+x^{51}+x^{52}+x^{56}+x^{57}+x^{58}+x^{59}+x^{63}+x^{69}+x^{70}+x^{71}+x^{72}+\\
&x^{77}+x^{80}+x^{81}+x^{82}+x^{84}+x^{86}+x^{87}+x^{90}+x^{92}+x^{94}+x^{95}+x^{96}+x^{101}+\\
&x^{106}+x^{107}+x^{108}+x^{109}+x^{111}+x^{112}+x^{113}+x^{115}+x^{116}+x^{120}+x^{121}+x^{123}+x^{128}+\\
&x^{130}+x^{132}+x^{133}+x^{135}+x^{138}+x^{140}+x^{142}+x^{143}+x^{144}+x^{146}+x^{148}+x^{150}+x^{156}+\\
&x^{163}+x^{164}+x^{166}+x^{167}+x^{168}+x^{170}+x^{171}+x^{172}+x^{175}+x^{176}+x^{178}+x^{179}+x^{181}+\\
&x^{182}+x^{183}+x^{184}+x^{186}+x^{189}+x^{190}+x^{191}+x^{192}+x^{195}+x^{199}+x^{201}+x^{202}+x^{203}+\\
&x^{205}+x^{207}+x^{213}+x^{214}+x^{216}+x^{217}+x^{220}+x^{221}+x^{222}+x^{223}+\\
&x^{225}+x^{226}+x^{227}+x^{228}+x^{230}+x^{231}+x^{233}+x^{235}+x^{237}+x^{238}+x^{240}+x^{242}+x^{245}+\\
&x^{249}+x^{250}+x^{253}+x^{255}+x^{256}+x^{259}+x^{260}
\end{aligned} \tag{11}$$

Here *degree* $\{g(x)\}$ =260 = $n - k$, $n = k + 260$ =4096 + 260 = 4356. The primitive BCH code parameters over $F_{2^{13}}$ are (8191, 8191 – 260) = (8191, 7931). In this case, $k = 4096 \Rightarrow$ 7931 – 4096 =3835. Therefore the parameters of the shortened BCH code are (8191 - 3835, 7931 - 3835) = (4356, 4096). Since (n – k) = 256 and degree $\{g(x)\}$ = 260, a t = 20 error correcting BCH code cannot be used with memory model.

Using (2) the probability of decoding error (which also constitutes the UBER) can be computed. We have made this computation for the BCH codes with t=16, t=17 and t=18 for a RBER of 10^{-6} (Memory model-I) in Table III and with t=18 and t=19 (Memory model-II) in Table IV. These values quantify the improvement in reliability brought about by the use of the BCH codes with block length k=1024 bytes and k=512 bytes respectively with $n-k \le 32$ bytes

TABLE III: Computed values of UBER for given RBER and *t* for memory model-I

t	RBER	UBER
16	10^{-6}	10^{-50}
17	10^{-6}	10^{-54}
18	10^{-6}	10^{-57}

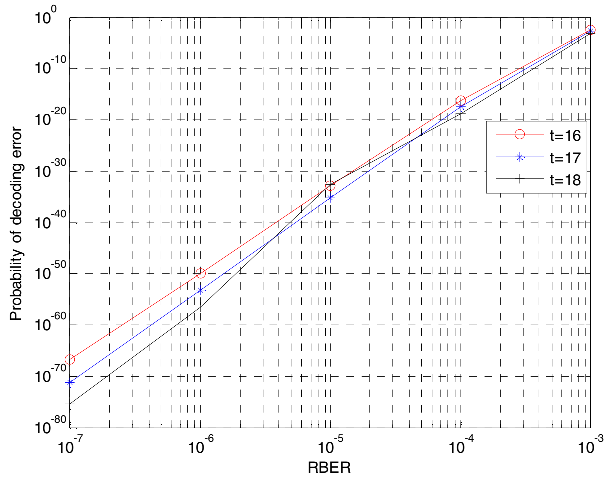

Figure 3. Performance of BCH codes with parameters *k*=1024 bytes, $n-k \leq 32$ bytes for different values of *t*

TABLE IV: Computed values of UBER for given RBER and *t* for memory model-II

t	RBER	UBER
18	10^{-6}	10^{-63}
19	10^{-6}	10^{-66}

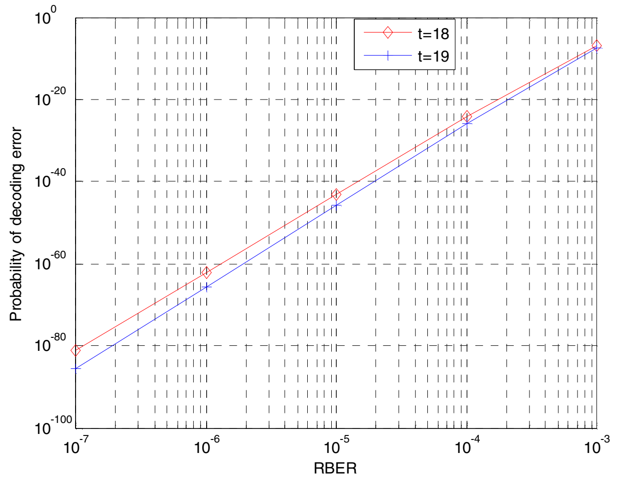

Figure 4. Performance of BCH codes with parameters *k*=512 bytes, $n-k \leq 32$ bytes for different values of *t*

V. CONCLUSION

We have synthesized solutions based on BCH codes capable of correcting a wide range of errors for application in Flash memories. These solutions have been worked out by adhering to the architectural constraints of flash memories. The choice of the code to be employed will depend on the RBER value encountered in the memory device and the degree of information integrity required by the end user (quantified by the value of target UBER). From the performance plots shown in Figures 3 we can infer that a RBER value of 10^{-6} is transformed by the *t*=18 BCH code (Memory Model I) to a UBER value of 10^{-57}. Similarly the *t*=19 BCH code (whose performance is sketched in Figure 4) transforms RBER values of 10^{-6} into UBER value of 10^{-66}. These values of UBER indicate very high degree of reliability and the flash memories protected by these codes should be suitable even in the most stringent data storage application.

REFERENCES

[1] Axel Mehnert, "Managing Flash memories with Intelligence", Industrial Embedded Systems E-Letter, March 2008.(www.industrial-embedded.com/articles/3009)

[2] Toru Tanzawa, Tomoharu Tanaka, Ken Takeuchi, Riichiri Shirota, Seiichi Aritome, Hiroshi Watanabe,Gertjan Hemink, Kazuuhiro Shimizu, Shinji Sato, Yuji Takeuchi and Kazunori Ohuchi, "A Compact On-Chip ECC for Low Cost Flash Memories", IEEE journal on Solid-State Circuits, vol.32, No.5, pp.662-668, May 1997.

[3] R. Micheloni, A.Marelli and R. Ravasio, Error Correction Code for Non-Volatile Memories, Springer 2008.

[4] Yuan Chen, "Flash Memory Reliability", NEPP 2008 Report, California Institute of Technology.

[5] Jim Cooke, "The Truth of NAND Flash Memory", Micron Technology Inc.(http://download.micron.com/pdf/presentation/events/WinHEC.Cooke.pdf)

[6] Neal Mielke, Todd Marquart, Ning Wu, Jeff Kessenich, Hanmant Belgal, Eric Schares, Falgun Trivedi, Evan Goodness and Leland R Nevill, "Bit Error Rate in Nand Flash Memories," Proc.46th Annual International Reliability Physics Symposium, pp.9-19, April 2008.

[7] F.Sun, S.Devarajan, K.Rose and T.Zhang, "Design of on-chip error correction systems for multilevel NOR and NAND flash memories," IET Circuits Devices Sysyems, vol 1, N0.3,pp.241-249, 2007.

[8] S.B.Wicker, Error Control Systems for Digital Communication and Storage, PHI, 1995.

[9] S.Lin and D.J.Costello, Error Control Coding, Pearson Education, 2003.

[10] Todd K. Moon, Error Correction Coding: Mathematical Methods and Algorithms, Wiley Interscience, 2006.

[11] E.Blahut, Algebraic Codes for Data Transmission, Cambridge University Press, 2003.

[12] Rajesh Shetty K, Sripati U, Prashantha Kumar H and B. Shankarananda, "Design and Construction of Algebrai Codes for Enhancing Data Integrity in Flash Memories", Intrnational journal on Advances in Communication Engineering, vol.2, No.2, ISSN:0975-6094, pp.51-56, July-Dec 2010..