International Journal of Electronics

ISSN: 0020-7217 (Print) 1362-3060 (Online) Journal homepage: https://www.tandfonline.com/loi/tetn20

High-speed and parallel approach for decoding of binary BCH codes with application to Flash memory devices

H. Prashantha Kumar , U. Sripati & K. Rajesh Shetty

To cite this article: H. Prashantha Kumar , U. Sripati & K. Rajesh Shetty (2012) High-speed and parallel approach for decoding of binary BCH codes with application to Flash memory devices, International Journal of Electronics, 99:5, 683-693, DOI: 10.1080/00207217.2011.643498

To link to this article: https://doi.org/10.1080/00207217.2011.643498

Published online: 21 Dec 2011.

International Journal of Electronics
Vol. 99, No. 5, May 2012, 683–693

Taylor & Francis
Taylor & Francis Group

High-speed and parallel approach for decoding of binary BCH codes with application to Flash memory devices

H. Prashantha Kumar*, U. Sripati and K. Rajesh Shetty

Electronics and Communication Engineering, National Institute of Technology Karnataka (NITK) Surathkal, Mangalore, India

(*Received 4 June 2011; final version received 30 September 2011*)

In this article, we propose a high-speed decoding algorithm for binary BCH codes that can correct up to 7 bits in error. Evaluation of the error-locator polynomial is the most complicated and time-consuming step in the decoding of a BCH code. We have derived equations for specifying the coefficients of the error-locator polynomial, which can form the basis for the development of a parallel architecture for the decoder. This approach has the advantage that all the coefficients of the error locator polynomial are computed in parallel (in one step). The roots of error-locator polynomial can be obtained by Chien's search and inverting these roots gives the error locations. This algorithm can be employed in any application where high-speed decoding of data encoded by a binary BCH code is required. One important application is in Flash memories where data integrity is preserved using a long, high-rate binary BCH code. We have synthesized generator polynomials for binary BCH codes (error-correcting capability, $t = 7$) that can be employed in Flash memory devices to improve the integrity of information storage. The proposed decoding algorithm can be used as an efficient, high-speed decoder in this important application.

Keywords: BCH codes; syndrome tree; error-locator polynomial; Flash memories

1. Introduction

The communication revolution and the advent of internet age have produced enormous demand for the increase in storage capacity and storage density. Physical/media improvements, along with sophisticated signal processing and coding techniques, have played a critical role in the constant augmentation of storage/communication channel capacities (Costello and Forney 2007). Every computer memory and data storage system has adopted some type of error-detection code or error-correction code in order to enhance system reliability. The reliability levels that are required by storage devices are extremely high. This is primarily because, unlike communication systems, no retransmission is generally possible. We expect to save our data and be able to retrieve it perfectly at any future time. Therefore, data integrity is a fundamental aspect of storage, security and reliability. As the recording density increases, 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 in the sense that the detection of an information bit is influenced by

*Corresponding author. Email: hprashanthakumar@gmail.com

ISSN 0020–7217 print/ISSN 1362–3060 online

http://dx.doi.org/10.1080/00207217.2011.643498
http://www.tandfonline.com

bits that are present in the recording medium in the immediate vicinity. This problem becomes more and more acute as the recording density increases. 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 (Micheloni, Marelli, and Ravasio 2008).

BCH and RS codes are amongst the most commonly used error correcting codes today and find application in wireless communications, high-speed modems and data storage systems. They belong to the class of cyclic codes and were independently invented by Hocquenghem in 1959 and Bose and Ray-Chaudhuri in 1960. The construction procedure adopted for specifying the generator polynomial of a BCH code specifies a lower bound on the minimum distance of the code. Two fields are involved in the construction of BCH codes. The coefficients of the generator polynomial (and hence all codewords) are drawn from the base (ground) field F_q. The roots of the generator polynomial lie in the extension field F_{q^m}. Decoding operation of a BCH code is relatively complicated (as compared to the encoding operation) because of the requirement that computational steps are performed in the larger field F_{q^m}. For binary BCH codes, $q = 2$.

Binary BCH codes are relatively simple and have good error-correcting performance, which is adequate in many applications. Many decoding algorithms have been proposed in literature. The Berlekamp–Massey and Sugiyama (Euclid's) algorithms are the most well-known and widely applied iterative techniques used to decode BCH codes. These enable us to determine the coefficients of the error-locator polynomial $\Lambda(x)$. An alternate solution to this problem involves the direct solution of a set of t equations in t unknowns where t is the error-correcting capability of the code. This second method can provide an extremely fast decoder that can be easily realized with modern VLSI technology (Kraft 1991). In this article, we have derived equations for calculating the coefficients of error-locator polynomial that can correct up to a maximum of 7 bits in error over the span of the codeword. These coefficients are specified in terms of the syndromes determined from the received codeword. Representing the syndromes in terms of a decision tree (syndrome tree) helps us to conclude whether errors have occurred and whether decoding is possible or not. Once we come to know that decoding is possible, we can substitute syndrome values in appropriate equations to evaluate the coefficients of the error-locator polynomial. In case decoding is not possible, decoding failure can be declared.

The motivation for this study has come from our earlier work on the synthesis of BCH codes for application in Flash memory systems where long high-rate BCH codes are currently employed to ensure data integrity (Rajesh Shetty, Sripati, Prashantha Kumar, and Shankarananda 2010). As Flash memories find increasing use in applications requiring high-speed data transfer, it is mandatory that internal operations of data search, encoding/decoding and memory access be completed in the shortest possible time. Thus, there is an urgent need to devise high-speed decoding algorithms employing parallel architecture (avoiding iterative steps in computation of $\Lambda(x)$) that can meet this requirement.

2. Synthesis of BCH codes for error protection in Flash memories

Use of error-control codes constitutes the most effective solution to improve data integrity of memory devices. In this study, we have synthesized the generator polynomial of a $t = 7$ error-correcting BCH code and derived equations to quantify the coefficients of the

error-locator polynomial. This facilitates the design of parallel high-speed decoders that can be used when this code is employed in Flash memory devices. Compared to the current standard (Mehnert 2008), where 6 bits in error can be corrected over a span of 4096 information bits, we propose a code that can correct seven errors. We now briefly discuss the organisation of Flash memories. As per Mehnert (2008), user memory is organized into blocks, pages and sectors. The smallest unit is a sector. Each sector has 512 bytes reserved for storing information and 16 bytes reserved for storing parity information. Table 1 gives the organisation of sectors, pages and blocks in 2-GB Flash memory.

The synthesis of a $t = 7$ BCH code for this application can be done with the help of the following steps. We assume that information block length k is equal to the number of bits stored in one sector (thus, $k = 512 \times 8 = 4096$ bits). Assuming that the BCH code is primitive and narrow sense, we are constrained to choose $n = 8191$, yielding a (8191, 4096) BCH code. However, given that only 128 bits are at most available for storing redundant information (Mehnert 2008), this is clearly not a suitable choice. The code will have to be shortened before it can be used in this application. Let α be the primitive element of the field $F_{2^{13}}$. From the BCH bound (Blahut 2003), the generator polynomial must have elements $\{\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7, \alpha^8, \alpha^9, \alpha^{10}, \alpha^{11}, \alpha^{12}, \alpha^{13}, \alpha^{14}\}$ as required roots. In addition, the requirement of conjugacy constraints imposes the condition that the generator polynomial also has all conjugates of these elements as roots (Wicker 1994). This is accomplished by defining the generator polynomial $g(x)$ as the least common multiple (LCM) of the minimal polynomials of these roots. To synthesize the generator polynomial of the shortened BCH code, we proceed as follows. Given $k = 4096, n - k \leq 128, t = 7$, the relevant minimal polynomials are listed in Table 2.

Thus, $g(x) = LCM\{M_1(x), M_3(x), M_5(x), M_7(x), M_9(x), M_{11}(x), M_{13}(x)\}$. Upon computation of the LCM of these polynomials, the generator polynomial is specified as expressed in (1).

$$\begin{aligned} g(x) = 1 &+ x^2 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{13} + x^{16} + x^{18} + x^{19} \\ &+ x^{23} + x^{25} + x^{26} + x^{29} + x^{30} + x^{31} + x^{32} + x^{33} + x^{35} \\ &+ x^{43} + x^{44} + x^{45} + x^{48} + x^{50} + x^{51} + x^{54} + x^{56} + x^{57} + x^{59} \\ &+ x^{61} + x^{62} + x^{67} + x^{75} + x^{91} \end{aligned} \tag{1}$$

We observe that the degree of this polynomial is equal to 91. Thus, $n - k = \deg(g(x)) = 91$. Since $n - k$ also corresponds to the number of redundant bits

Table 1. Memory organisation of a 2-GB Flash memory device.

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

Source: Mehnert (2008).

Table 2. List of minimal polynomials of the required roots for BCH code with $t = 7$.

$M_1(x) = 1 + x + x^3 + x^4 + x^{13}$
$M_3(x) = 1 + x + x^4 + x^5 + x^7 + x^9 + x^{10} + x^{13}$
$M_5(x) = 1 + x + x^4 + x^7 + x^8 + x^{11} + x^{13}$
$M_7(x) = 1 + x + x^2 + x^3 + x^6 + x^8 + x^9 + x^{10} + x^{13}$
$M_9(x) = 1 + x^5 + x^6 + x^7 + x^8 + x^{12} + x^{13}$
$M_{11}(x) = 1 + x + x^5 + x^7 + x^8 + x^9 + x^{13}$
$M_{13}(x) = 1 + x^3 + x^4 + x^5 + x^6 + x^{12} + x^{13}$

in the codeword, it is clear that this redundancy requirement can be accommodated by the Flash memory organisation. This generator polynomial defines a (8191, 8100) binary primitive length BCH code. Since, the information block in this case consists of only 4096 bits, the code can be shortened to a (4187, 4096) shortened BCH code. In the remaining part of this article, we have derived equations to compute the coefficients of $\Lambda(x)$ using a non-iterative procedure which facilitates fast parallel decoding.

3. Decoding of binary BCH codes

The algebraic decoding of binary BCH codes has the following general steps (Moon 2005):

- Computation of the syndromes $S_j, 1 \leq j \leq 2t$ from the received word. S_j is computed as $S_j = r(\alpha^j)$, where α^j is a root of generator polynomial $g(x)$.
- Use of an iterative algorithm (Berlekamp–Massey, Sugiyama) or direct solution (Peterson) to calculate $\Lambda(x)$.
- Determination of the roots of $\Lambda(x)$ using Chien's search algorithm, which is an exhaustive search over all the elements in the appropriate finite field. The reciprocals (inverses) of the roots represent error locations. Since the code is binary, flipping the erroneous bits gives the correct codeword.

The computation of the error-locator polynomial is the step that requires the highest computational resources and time. Our aim is to propose a straightforward non-iterative solution to this problem. In this section, we have presented a solution to the problem of computing coefficients of $\Lambda(x)$ for BCH codes characterized by error-correcting capability $t = 7$ over the span of the codeword. From the equations, it can be seen that all the necessary computational steps can be performed using simple combinational circuits. As a result, decoding of the received codeword can be accomplished in a time span much less than what would be needed if we had employed contemporary iterative decoding techniques.

Suppose that a codeword $\mathbf{c}(x) = c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1}$ is transmitted. The received polynomial (comprising a $c(x)$ possibly corrupted by additive noise) is specified by,

$$\mathbf{r}(x) = r_0 + r_1x + r_2x^2 + \cdots + r_{n-1}x^{n-1} \tag{2}$$

Let $\mathbf{e}(x)$ denote the error pattern introduced by the channel. Then

$$\mathbf{r}(x) = \mathbf{c}(x) + \mathbf{e}(x) \tag{3}$$

For $1 \leq j \leq 2t$, the jth component of the syndrome is given by

$$S_j = \mathbf{r}(\alpha^j) = \mathbf{e}(\alpha^j) = r_0 + r_1\alpha^j + r_2\alpha^{2j} + r_3\alpha^{3j} + \cdots + r_{n-1}\alpha^{(n-1)j} \tag{4}$$

Let us assume that the received word $\mathbf{r}$ has υ errors ($\upsilon \leq t$) in positions $i_1, i_2, \ldots, i_\upsilon$. Since the code is binary, the errors in these positions have values $e_{ij} = 1$. Therefore, syndrome sequence can be re-expressed in terms of error locations as Lin and Costello (2004)

$$S_j = \sum_{l=1}^{\upsilon} e_{i_l}(\alpha^j)^{i_l} = \sum_{l=1}^{\upsilon} (\alpha^{i_l})^j = \sum_{l=1}^{\upsilon} X_l^j, \quad j = 1, 2, \ldots, 2t \tag{5}$$

The $\{X_l\}$ represent error locations which indicate the positions of the errors in the received codeword. Expanding (5), for $j = 1, 2, \ldots, 2t$, we obtain a sequence of $2t$ syndrome equations in the υ unknown error locations.

$$\left.\begin{aligned} S_1 &= X_1 + X_2 + \ldots + X_\upsilon \\ S_2 &= X_1^2 + X_2^2 + \ldots + X_\upsilon^2 \\ S_3 &= X_1^3 + X_2^3 + \ldots + X_\upsilon^3 \\ &\vdots \\ S_{2t} &= X_1^{2t} + X_2^{2t} + \ldots + X_\upsilon^{2t} \end{aligned}\right\} \tag{6}$$

The above set of equations forms a set of nonlinear algebraic equations in multiple variables and is called power-sum symmetric functions (Lin and Costello 2004; Moon 2005). Rather than attempting to solve these nonlinear equations directly, let us define a new polynomial called *error-locator polynomial*, which casts the problem in a different, and more tractable, setting. The error-locator polynomial is defined as

$$\Lambda(x) = \prod_{l=1}^{\upsilon} (1 - X_l x) = \Lambda_0 + \Lambda_1 x + \Lambda_2 x^2 + \cdots + \Lambda_{\upsilon-1} x^{\upsilon-1} + \Lambda_\upsilon x^\upsilon \tag{7}$$

where $\Lambda_0 = 1$. Once we knew the coefficients of $\Lambda(x)$, we could find the zeros of $\Lambda(x)$ to obtain the error locations. The roots of the error-locator polynomial are the reciprocals of the error locators.

One key observation here is that there is a linear relationship between the syndromes and coefficients of the error-locator polynomial. This relationship is described by Newton's identities, which can be applied for any field (Reed and Chen 1999). From (6), we can write the following relationship between syndromes and the coefficients of error-locator polynomial using Newton identities.

$$\left.\begin{aligned} S_k + \Lambda_1 S_{k-1} + \cdots + \Lambda_{k-1} S_1 + k\Lambda_k = 0 \quad 1 \leq k \leq \upsilon \\ S_k + \Lambda_1 S_{k-1} + \cdots + \Lambda_{\upsilon-1} S_{k-\upsilon+1} + \Lambda_\upsilon S_{k-\upsilon} = 0 \quad k > \upsilon \end{aligned}\right\} \tag{8}$$

Newton's identities are linear in the υ unknown coefficients of the error-locator polynomial. In a binary field (F_2), the above system of equations can be greatly simplified. First, we note that $j\Lambda_j = \Lambda_j$ if j is odd, and $j\Lambda_j = 0$ if j is even. Furthermore, $S_{2j} = S_j^2$.

We can thus write Newton's identities (8) as

$$\left.\begin{aligned} &S_1 + \Lambda_1 = 0 \\ &S_3 + \Lambda_1 S_2 + \Lambda_2 S_1 + \Lambda_3 = 0 \\ &S_5 + \Lambda_1 S_4 + \Lambda_2 S_3 + \Lambda_3 S_2 + \Lambda_4 S_1 + \Lambda_5 = 0 \\ &\qquad\qquad \vdots \\ &S_{2t-1} + \Lambda_1 S_{2t-2} + \Lambda_2 S_{2t-3} + \cdots + \Lambda_t S_{t-1} = 0 \end{aligned}\right\} \tag{9}$$

In the following section, we give the closed-form solutions for solving (9) to find the coefficients of error-locator polynomial for $t = 7$ errors using Peterson's direct solution decoding algorithm. The solution for the case of $t \leq 5$ has been discussed in Michelson and Levesque (1985). The solution for the case $t = 6$ has been worked out, but it has not been presented here for want of space.

4. Direct-solution decoding for BCH codes characterized by $t = 7$

Equation (9) can be expressed in matrix form as follows.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ S_2 & S_1 & 1 & 0 & \cdots & 0 & 0 \\ S_4 & S_3 & S_2 & S_1 & \cdots & 0 & 0 \\ S_6 & S_5 & S_4 & S_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ S_{2t-4} & S_{2t-5} & S_{2t-6} & S_{2t-7} & \cdots & S_{t-2} & S_{t-3} \\ S_{2t-2} & S_{2t-3} & S_{2t-4} & S_{2t-5} & \cdots & S_t & S_{t-1} \end{bmatrix} \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \\ \Lambda_4 \\ \vdots \\ \Lambda_{t-1} \\ \Lambda_t \end{bmatrix} = \begin{bmatrix} S_1 \\ S_3 \\ S_5 \\ S_7 \\ \vdots \\ S_{2t-3} \\ S_{2t-1} \end{bmatrix} \tag{10}$$

This matrix equation is described by $\mathbf{A}\Lambda = \mathbf{S}$, where $\mathbf{A}$ represents the matrix of syndromes, Λ represents the column vector of coefficients of $\Lambda(x)$ and $\mathbf{S}$ represents the column vector of syndromes. If the number of errors introduced by the channel is exactly t, then $\mathbf{A}$ is non-singular and the system in (10) has a unique solution. If fewer than t errors have occurred, we proceed by eliminating the two bottom rows and the two rightmost columns of $\mathbf{A}$ and check to see if the resulting matrix is non-singular (Moon 2005). This process is continued until we obtain a reduced matrix which is non-singular. This matrix-based approach used to determine the coefficients of $\Lambda(x)$ is called *Peterson's algorithm* for decoding binary BCH codes. Michelson and Levesque (1985) have provided explicit formulae for computing the coefficients of $\Lambda(x)$ for values of $t \leq 5$. We present below the closed-form expressions to determine coefficients of $\Lambda(x)$ for binary BCH codes characterized by error-correcting capability $t = 7$.

For seven error-correcting binary BCH codes, we can write (10) as

$$\begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \\ \Lambda_4 \\ \Lambda_5 \\ \Lambda_6 \\ \Lambda_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ S_1^2 & S_1 & 1 & 0 & 0 & 0 & 0 \\ S_1^4 & S_3 & S_1^2 & S_1 & 1 & 0 & 0 \\ S_3^2 & S_5 & S_1^4 & S_3 & S_1^2 & S_1 & 1 \\ S_1^8 & S_7 & S_3^2 & S_5 & S_1^4 & S_3 & S_1^2 \\ S_5^2 & S_9 & S_1^8 & S_7 & S_3^2 & S_5 & S_1^4 \\ S_3^4 & S_{11} & S_5^2 & S_9 & S_1^8 & S_7 & S_3^2 \end{bmatrix}^{-1} \begin{bmatrix} S_1 \\ S_3 \\ S_5 \\ S_7 \\ S_9 \\ S_{11} \\ S_{13} \end{bmatrix} \tag{11}$$

This system of equations when solved (and simplified using Horner's rule) yields,

$$\Lambda_1 = S_1$$

$$\Lambda_2 = \frac{S_9S_7^2 + (S_7S_{11} + S_9^2 + S_{13}S_5)S_5 + (S_5^4 + S_{13}S_7 + S_9S_{11} + (S_7S_5^2 + (S_5S_9 + S_{11}S_3)S_3)S_3)S_3 + (S_7S_3^3 + (S_5^2S_9 + (S_{11}S_5 + (S_{13} + (S_5^2 + S_7S_3)S_3)S_3)S_3)S_3 + (S_7^3 + S_9S_5S_7 + (S_7S_{11} + S_{13}S_5 + (S_5^3 + S_7S_3S_5)S_3)S_3 + (S_5^4 + S_{13}S_7 + S_9S_{11} + (S_7S_5^2 + (S_{11}S_3 + S_7^2)S_3)S_3 + ((S_7^2 + S_5S_9)S_5 + S_3^3S_5^2 + (S_{13}S_5 + S_9^2 + (S_5S_7 + S_9S_3)S_3^2 + ((S_{11}S_3 + S_7^2)S_3 + (S_9S_7 + (S_{13} + (S_5^2 + S_7S_3)S_3)S_3 + ((S_5S_7 + S_3^4)S_3 + (S_{11}S_3 + S_7^2 + (S_3S_5^2 + S_{13} + (S_{11} + S_3S_5^2 + (S_5^2 + S_7S_3 + (S_3^3 + S_9 + S_3S_1^6)S_1)S_1^2)S_1)S_1)S_1)S_1)S_1)S_1)S_1}{\Delta}$$

$$\Lambda_3 = \frac{(S_5^2S_{11} + S_7^3 + (S_7S_{11} + S_9^2 + (S_5^3 + (S_5S_7 + S_3^4)S_3)S_3)S_3 + (S_9S_7^2 + (S_7S_{11} + S_9^2 + S_{13}S_5)S_5 + (S_{13}S_7 + S_9S_{11} + (S_7^2 + S_5S_3^3 + S_5S_9)S_3^2)S_3 + (S_7S_5^3 + (S_5S_7^2 + (S_9S_7 + (S_3S_5^2 + S_{13})S_3)S_3)S_3 + ((S_9S_7 + S_{11}S_5)S_5 + (S_7S_{11} + S_{13}S_5 + (S_5^3 + (S_5S_7 + (S_3^3 + S_9)S_3)S_3)S_3)S_3 + (S_{13}S_7 + S_9S_{11} + (S_7^2 + S_5S_9)S_3^2 + (S_7S_{11} + S_{13}S_5 + (S_5S_7 + S_3^4)S_3^2 + (S_{11}S_5 + (S_3^2S_7 + S_{13})S_3 + (S_5^3 + (S_5S_7 + (S_3^3 + S_9)S_3)S_3 + (S_7^2 + S_5S_9 + (S_{11} + S_5S_3^2)S_3 + (S_3^2S_7 + S_{13} + (S_5S_7 + S_3^4 + (S_3S_5 + (S_7 + (S_3^2 + S_1^6)S_1)S_1^4)S_1)S_1)S_1)S_1^2)S_1^2)S_1)S_1)S_1}{\Delta}$$

$$\Lambda_4 = \frac{(S_7S_{11} + S_9^2)S_7 + (S_{13}S_7 + S_9S_{11})S_5 + (S_{13}S_9 + S_7S_5^3 + S_{11}^2 + (S_9S_7 + S_{11}S_5 + (S_3S_5^2 + S_{13})S_3)S_3^2)S_3 + ((S_7^2 + S_5S_9)S_5^2 + (S_{13}S_5 + S_9^2 + (S_5S_7 + S_3^4)S_3^2)S_3^2 + (S_9S_7^2 + S_{13}S_5^2 + (S_{11} + S_5S_3^2)S_3^4 + (S_{13}S_9 + S_7S_5^3 + S_{11}^2 + (S_9S_7 + S_{11}S_5 + S_3^2S_5^2)S_3^2 + (S_7^3 + S_9S_5S_7 + (S_7S_{11} + S_{13}S_5 + (S_5S_7 + (S_3^3 + S_9)S_3)S_3^2)S_3 + (S_{13}S_7 + S_9S_{11} + (S_7^2 + S_5S_9)S_3^2 + (S_{13}S_3^2 + S_5S_7^2 + (S_7S_{11} + S_{13}S_5 + (S_5S_7 + S_3^4)S_3^2 + (S_9S_7 + S_{11}S_5 + (S_5^3 + S_7^2S_9 + (S_5S_7 + S_3^4 + (S_{11} + S_5S_3^2 + (S_9 + S_5S_1^4)S_1^2)S_1)S_1^2)S_1)S_1)S_1)S_1)S_1}{\Delta}$$

$$\Lambda_5 = \frac{(S_5^2S_{11} + S_7^3)S_5 + (S_9S_7^2 + S_{13}S_5^2 + (S_{13}S_7 + S_9S_{11} + (S_5S_9 + (S_{11} + S_5S_3^2)S_3)S_3^2)S_3)S_3 + ((S_7S_{11} + S_9^2)S_7 + (S_5^4 + S_{13}S_7 + S_9S_{11})S_5 + (S_{13}S_9 + S_7S_5^3 + S_{11}^2 + ((S_7^2 + S_5S_9)S_5 + (S_9S_7 + S_{11}S_5 + (S_5^2 + S_7S_3)S_3^2)S_3)S_3)S_3 + (S_{13}S_9 + S_{11}^2 + (S_5S_7^2 + (S_{11}S_5 + S_{13}S_3)S_3)S_3)S_3 + ((S_9S_7 + S_{11}S_5)S_5 + (S_7S_{11} + S_{13}S_5 + (S_5^3 + (S_5S_7 + (S_3^3 + S_9)S_3)S_3)S_3)S_3 + (S_7S_5^2 + S_3^2S_{11} + (S_7^2 + S_5S_9 + (S_{11} + S_5S_3^2)S_3 + (S_3^2S_7 + S_{13} + (S_5^2 + S_3^2S_1^4)S_1^3)S_1)S_1^3)S_1)S_1^3)S_1}{\Delta}$$

$$\Lambda_6 = \frac{S_9^3 + S_{13}S_7^2 + (S_{13}S_9 + S_{11}^2)S_5 + ((S_7^2 + S_5S_9)S_5^2 + (S_7^3 + S_9S_5S_7 + (S_{13}S_5 + S_9^2 + (S_5S_7 + (S_3^3 + S_9)S_3)S_3^2)S_3)S_3)S_3 + ((S_7S_{11} + S_9^2)S_7 + (S_5^4 + S_{13}S_7 + S_9S_{11})S_5 + (S_{13}S_9 + S_7S_5^3 + S_{11}^2 + ((S_7^2 + S_5S_9)S_5 + (S_9S_7 + S_{11}S_5 + (S_5^2 + S_7S_3)S_3^2)S_3)S_3)S_3 + (S_9S_7^2 + (S_7S_{11} + S_9^2 + S_{13}S_5)S_5 + (S_{13}S_7 + S_9S_{11} + (S_7^2 + S_5S_9 + S_3^3 + S_5S_9)S_3^2)S_3 + ((S_9S_7 + S_{11}S_5)S_5 + (S_7S_{11} + S_{13}S_5 + (S_5^3 + (S_5S_7 + (S_3^3 + S_9)S_3)S_3)S_3)S_3 + (S_5S_7^2 + (S_9S_7 + S_{11}S_5 + S_{13}S_3)S_3 + (S_7S_5^2 + S_3^2S_{11} + (S_5^3 + (S_5S_7 + (S_3^3 + S_9)S_3)S_3 + (S_3^2S_7 + S_{13} + (S_{11} + S_7S_1^4)S_1^2)S_1^2)S_1^2)S_1^2)S_1^2)S_1^2}{\Delta}$$

$$\Lambda_7 =$$

$$S_7^4 + (S_9S_7^2 + (S_{13}S_5 + S_9^2)S_3)S_5 + (S_5^5 + (S_{13}S_9 + S_7S_5^3 + S_{11}^2 + ((S_7^2 + S_5S_9)S_5 + (S_9S_7 + (S_{13} + (S_5^2 + S_7S_3)S_3)S_3)S_3)S_3)S_3 + (S_9^3 + S_{13}S_7^2 + (S_{13}S_9 + S_{11}^2)S_5 + ((S_7^2 + S_5S_9)S_5^2 + (S_7^3 + S_9S_5S_7 + (S_{13}S_5 + S_9^2 + (S_5S_7 + (S_5^3 + S_9)S_3)S_3^2)S_3)S_3)S_3 + ((S_5^2S_{11} + S_7^3)S_5 + (S_9S_7^2 + S_{13}S_5^2 + + (S_{13}S_7 + S_9S_{11} + (S_5S_9 + (S_{11} + S_5S_3^2)S_3)S_3^2)S_3)S_3 + ((S_7S_{11} + S_9^2)S_7 + (S_5^4 + S_{13}S_7 + S_9S_{11})S_5 + (S_{13}S_9 + S_7S_5^3 + S_{11}^2 + ((S_7^2 + S_5S_9)S_5 + (S_9S_7 + S_{11}S_5 + (S_5^2 + S_7S_3)S_3^2)S_3)S_3)S_3 + ((S_7^2 + S_5S_9)S_5^2 + (S_5^2S_{11} + S_7^3 + (S_7S_{11} + S_{13}S_5 + S_3S_5^3)S_3)S_3 + (S_9S_7^2 + (S_7S_{11} + S_9^2 + S_{13}S_5)S_5 + (S_{13}S_7 + S_9S_{11} + (S_7^2 + S_5S_3^3 + S_5S_9)S_3^2)S_3 + (S_{13}S_9 + S_{11}^2 + (S_5S_7^2 + (S_{11}S_5 + S_{13}S_3)S_3)S_3 + ((S_9S_7 + S_{11}S_5)S_5 + (S_7S_{11} + S_{13}S_5 + (S_5^3 + (S_5S_7 + (S_3^3 + S_9)S_3)S_3)S_3 + (S_{13}S_7 + S_9S_{11} + (S_7S_5^2 + S_3^2S_{11})S_3 + (S_5S_7^2 + (S_9S_7 + S_{11}S_5 + S_{13}S_3)S_3 + (S_7S_{11} + S_9^2 + (S_5^3 + (S_5S_7 + (S_3^3 + S_9)S_3)S_3)S_3 + (S_7S_5^2 + S_3^2S_{11} + (S_9S_7 + (S_3^2S_7 + S_{13})S_3 + (S_5^3 + (S_5S_7 + (S_3^3 + S_9)S_3)S_3 + (S_7^2 + S_5S_9 + (S_{11} + S_5S_3^2)S_3 + (S_3^2S_7 + S_{13} + (S_5S_7 + S_9S_3 + (S_{11} + (S_5^2 + (S_3S_5 + (S_7 + (S_3^2 + S_1^6)S_1)S_1^2)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1 \Big\} 12$$

$$\Delta$$

where common denominator Δ is given by

$$\Delta =$$

$$S_5^2S_{11} + S_7^3 + (S_7S_{11} + S_9^2 + (S_5^3 + (S_5S_7 + S_3^4)S_3)S_3)S_3 + (S_5^4 + (S_7S_5^2 + (S_7^2 + (S_{11} + S_5S_3^2)S_3)S_3)S_3 + ((S_7^2 + S_5S_9)S_5 + (S_9S_7 + S_{11}S_5 + S_7S_3^3)S_3 + (S_7S_{11} + S_9^2 + (S_5^3 + (S_5S_7 + (S_3^3 + S_9)S_3)S_3)S_3 + (S_7S_5^2 + S_3^2S_{11} + (S_9S_7 + (S_3^2S_7 + S_{13})S_3 + (S_5^3 + (S_5S_7 + (S_3^3 + S_9)S_3)S_3 + (S_7^2 + S_5S_9 + (S_{11} + S_5S_3^2)S_3 + (S_3^2S_7 + S_{13} + (S_5S_7 + S_9S_3 + (S_{11} + (S_5^2 + (S_3S_5 + (S_7 + (S_3^2 + S_1^6)S_1)S_1^2)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1)S_1$$

We can observe that the coefficients of error-locator polynomials are expressed in terms of odd syndromes. By assuming each odd syndrome to be either zero or not zero, we can derive the syndrome decision tree as shown in Figure 1. In this diagram, A corresponds to the no-error condition, B to the possibility of the channel having introduced an uncorrectable error pattern and C to the channel having introduced a correctable error pattern. D points to the channel having introduced an error pattern of weight six. From an inspection of the tree, it is observed that leaf node B is reached whenever one of the following conditions is satisfied:

(1) $S_{11} \neq 0, S_1 - S_9 = 0$
(2) $S_{11} = 0, S_9 \neq 0, S_1 - S_7 = 0$
(3) $S_{11} \neq 0, S_9 \neq 0, S_1 - S_7 = 0$

Whenever one of these conditions is satisfied, it will result in the solution to the matrix equation taking on $\frac{0}{0}$ form and hence a solution to the system of equations does not exist.

In a similar manner, leaf node D is reached whenever one of the following two conditions is satisfied.

(1) $S_{11} = S_9 = S_7 = S_3 = S_1 = 0, S_5 \neq 0$
(2) $S_{11} = S_7 = S_3 = S_1 = 0, S_9 \neq 0, S_5 \neq 0$

Careful inspection of the **A** matrix under these conditions reveals that it is singular. However, with the deletion of the bottom-most row and right-most column, the reduced matrix is non-singular. This implies that the number of errors introduced by the channel is six. In example 1, we have shown that if the channel introduces an error pattern, the decoding algorithm can successfully compute the coefficients of the error-locator polynomial from which the error locations can be correctly determined.

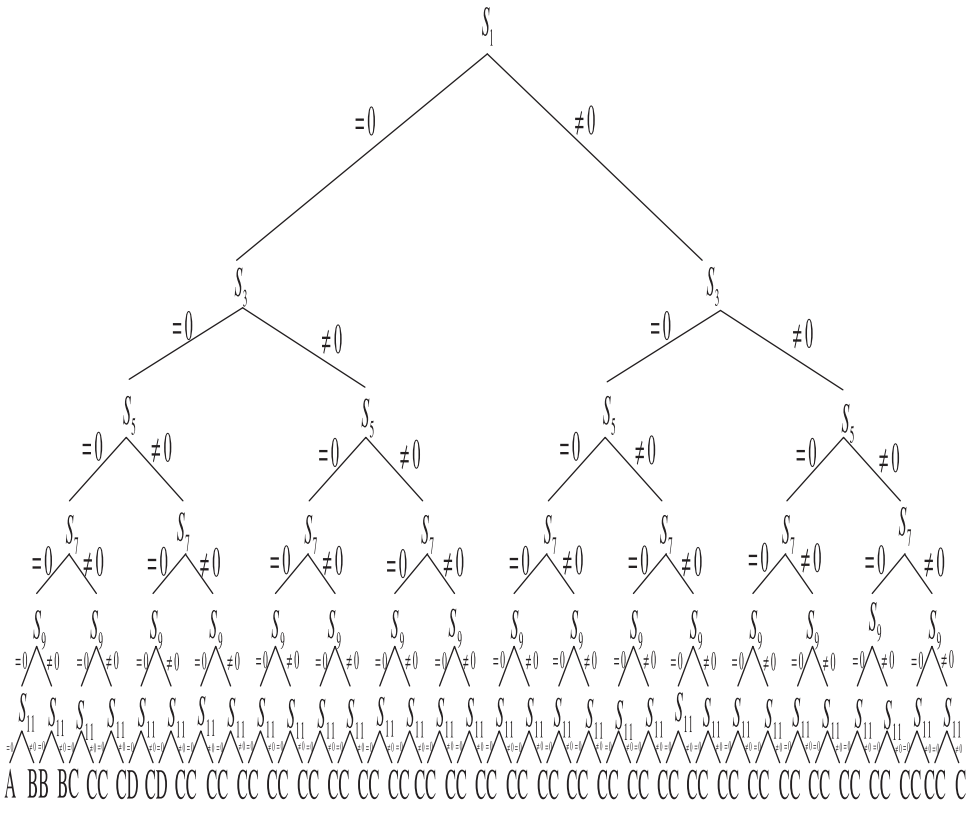

Figure 1. Syndrome tree for seven error-correcting binary BCH codes.

Example 1: In this example, the system of equations in (12) is verified by cross-checking the results with Berlekamp–Massey algorithm. We have considered the primitive length (63, 24) seven-error-correcting binary BCH code. Let us assume that the all-zero vector **c** of length 63 is transmitted and received vector is $\mathbf{r} = (111111100\cdots000)$, which has seven errors. The syndromes are calculated using (4). Since this is a binary BCH code, we have to calculate only t odd syndromes. The syndromes are $S_1 = \alpha^{20}$, $S_3 = \alpha^{10}$, $S_5 = \alpha^{14}$, $S_7 = \alpha^{12}$, $S_9 = 0$, $S_{11} = \alpha^{27}$ and $S_{13} = \alpha^{6}$. From the equations derived for coefficients of error-locator polynomial given in (12), we can calculate

$$\Lambda_1 = S_1 = \alpha^{20}$$

$$\Lambda_2 = \frac{\alpha^{25}}{\alpha^{15}} = \alpha^{10}$$

$$\Lambda_3 = \frac{\alpha^{57}}{\alpha^{15}} = \alpha^{42}$$

$$\Lambda_4 = \frac{\alpha^{60}}{\alpha^{15}} = \alpha^{45}$$

$$\Lambda_5 = \frac{\alpha^{34}}{\alpha^{15}} = \alpha^{19}$$

Table 3. Berlekamp–Massey algorithm for a (63, 24) seven error correcting BCH code.

Iteration number, μ	Error locator polynomial $\Lambda^{(\mu)}(x)$ in each μth iteration
1	$1+\alpha^{20}x$
2	$1+\alpha^{20}x+\alpha^{12}x^2$
3	$1+\alpha^{20}x+\alpha^{36}x^2+\alpha^{36}x^3$
4	$1+\alpha^{20}x+\alpha^{33}x^2+\alpha^{11}x^3+\alpha^{14}x^4$
5	$1+\alpha^{20}x+\alpha^{48}x^2+\alpha^{23}x^3+\alpha^{37}x^4+\alpha^{29}x^5$
6	$1+\alpha^{20}x+\alpha^{19}x^2+\alpha^{58}x^3+\alpha^{39}x^4+\alpha^{39}x^5+\alpha^{30}x^6$
7	$1+\alpha^{20}x+\alpha^{10}x^2+\alpha^{42}x^3+\alpha^{45}x^4+\alpha^{19}x^5+\alpha^{35}x^6+\alpha^{21}x^7$

$$\Lambda_6 = \frac{\alpha^{50}}{\alpha^{15}} = \alpha^{35}$$

$$\Lambda_7 = \frac{\alpha^{36}}{\alpha^{15}} = \alpha^{21}$$

Therefore, $\Lambda(x) = 1+\alpha^{20}x+\alpha^{10}x^2+\alpha^{42}x^3+\alpha^{45}x^4+\alpha^{19}x^5+\alpha^{35}x^6+\alpha^{21}x^7$.

The same problem has been worked out with Berlekamp–Massy algorithm for validation. These iterative steps are given in Table 3.

Thus, we see that the Berlekamp–Massey algorithm and the equations derived in this article yield the same values for the coefficients of error-locator polynomial when the same received vector **r** is input.

5. Conclusion

We have derived generalized equations for computing the coefficients of error-locator polynomials for binary BCH codes that can correct up to seven errors. A survey of literature reveals that direct computation of coefficients of the error-locator polynomial has been worked out only for the case of five or fewer errors (Michelson and Levesque 1985). We have also derived the generator polynomial of a (4187, 4096) BCH code that can correct up to 7 bits in error. This code can find application in protecting information integrity in Flash memory systems. The main advantage of this approach when compared to decoding BCH codes is design simplicity (due to use of combinational logic circuits) and speed (due to the avoidance of iteration). Computation can be carried out by simple combinational logic and ROM implemented look-up tables. This algorithm can also be advantageously employed in wireless applications employing BCH codes for error protection, in which high-speed/low-complexity decoding is essential.

References

Blahut, R.E. (2003), *Algebraic Codes for Data Transmission* (1st ed.), Cambridge: Cambridge University Press.

Costello, D., and Forney, D. (2007), 'Channel Coding: The Road to Channel Capacity', *Proceedings of the IEEE*, 95, 1150–1177.

Kraft, C. (1991), 'Closed Solution of Berlekamp's Algorithm for Fast Decoding of BCH Codes', *IEEE Transactions on Communications*, 39, 1721–1725.

Lin, S., and Costello, D.J. (2004), *Error Control Coding* (2nd ed.), Englewood Cliffs, NJ: Prentice Hall.

Mehnert, A. (2008), 'Managing Flash Memories with Intelligence', Industrial Embedded Systems E-Letter, http://industrial-embedded.com/managing-flash-memory-intelligence.

Micheloni, R., Marelli, A., and Ravasio, R. (2008), *Error Correction Codes for Non-Volatile Memories*, Heidelberg, Berlin: Springer.

Michelson, M., and Levesque, A.H. (1985), *Error-Control Techniques for Digital Communication* (1st ed.), New York, NY: Wiley - Inter Science.

Moon, T.K. (2005), *Error Correction Coding: Mathematical Methods and Algorithms* (1st ed.), Hoboken, NJ: Wiley-Inter Science.

Rajesh Shetty, K., Sripati, U., Prashantha Kumar, H., and Shankarananda, B. (2010), 'Synthesis of BCH Codes for Enhancing Data Integrity in Flash Memories', in *Proceedings of the 5th International Conference on Industrial and Information Systems, ICIIS 2010*, Mangalore, India, 29 July – 1 August 2010, pp. 119–124.

Reed, I.S., and Chen, X. (1999), *Error-Control Coding for Data Networks* (1st ed.), Norwell, MA: Kluwer Academic Press.

Wicker, S.B. (1994), *Error Control Systems for Digital Communication and Storage*, Upper Saddle River, NJ: Prentice Hall.