Chapter 6
BCH Codes
Outline

- Description of the Codes
- Decoding of the BCH Codes
- Implementation of Galois Field Arithmetic
- Implementation of Error Correction
- Nonbinary BCH Codes and Reed-Solomon Codes
- Weight Distribution and Error Detection of Binary BCH Codes
The Bose, Chaudhuri, and Hocquenghem (BCH) codes form a large class of powerful random error-correcting cyclic codes. This class of codes is a remarkable generalization of the Hamming codes for multiple-error correction. Binary BCH codes were discovered by Hocquenghem in 1959 and independently by Bose and Chaudhuri in 1960. Generalization of the binary BCH codes to codes in $p^m$ symbols (where $p$ is a prime) was obtained by Gorenstein and Zierler. Among the nonbinary BCH codes, the most important subclass is the class of Reed-Solomon (RS) codes. Among all the decoding algorithms for BCH codes, Berlekamp’s iterative algorithm, and Chien’s search algorithm are the most efficient ones.
Description of the Codes
Description of the Codes

∀ integer \( m \geq 3, \ t < 2^{m-1} \), \( \exists \) a binary BCH codes with

\[
n = 2^m - 1 \rightarrow \text{Block length}
\]

\[
n - k \leq mt \rightarrow \text{Number of parity-check digits}
\]

\[
d_{\min} \geq 2t + 1 \rightarrow \text{Minimum distance}
\]

Clearly, this code is capable of correcting any combination of \( t \) or fewer errors in a block of \( n = 2^m - 1 \) digits. We call this code a \textit{t-error-correcting} BCH code. The generator polynomial of this code is specified in terms of its roots from the Galois field \( GF(2^m) \).

Let \( \alpha \) be a primitive element of \( GF(2^m) \). The generator poly. \( g(x) \) of the \( t \)-error-correcting BCH code of length \( 2^m - 1 \) is the lowest-degree poly. over \( GF(2) \) which has

\[\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}\] as its roots.
Description of the Codes

It follows from **Theorem 2.7** that \( g(x) \) has \( \alpha, \alpha^2, ..., \alpha^{2t} \) and their conjugates as all its roots. Let \( \phi_i(x) \) be the **minimal poly.** of \( \alpha^i \). Then \( g(x) \) must be the **least common multiple** of \( \phi_1(x), \phi_2(x), ..., \phi_{2t}(x) \), that is,

\[
g(x) = \text{LCM}\{\phi_1(x), \phi_2(x), ... \phi_{2t}(x)\}
\]

If \( i \) is an even integer, it can be expressed as a product of the following form:

\[
i = i'2^l,
\]

where \( i' \) is an odd number and \( l \geq 1 \). Then \( \alpha^i = (\alpha^{i'})^{2^l} \) is a conjugate of \( \alpha^i \) and therefore \( \alpha^i \) and \( \alpha^{i'} \) have the same minimal poly., that is,

\[
\phi_i(x) = \phi_{i'}(x).
\]
Hence, even power of $\alpha$ has the same minimal poly. as some preceding odd power of $\alpha$.

$$g(x) = \text{LCM} \left\{ \phi_1(x), \phi_3(x), \ldots \phi_{2t-1}(x) \right\}$$

$$\deg [\phi_i(x)] \leq m \quad (\text{Theorem 2.15})$$

$$\therefore \deg [g(x)] \leq mt \quad \therefore n - k \leq mt$$

The BCH codes defined above are usually called primitive (or narrow-sense) BCH codes.
The single-error-correcting BCH codes of length $2^m - 1$ is generated by $g(x) = \phi_1(x)$ since $t = 1$.

$\therefore \alpha$ is a primitive element of $GF(2^m)$

$\therefore \phi_1(x)$ is a primitive poly. of degree $m$

$(\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{2^m} = 1)$

$\therefore$ the single-error-correcting BCH codes of length $2^m - 1$ is a Hamming code

Ex 6.1

$\alpha$ is a primitive element of $GF(2^4)$ given by Table 2.8 such that $1 + \alpha + \alpha^4 = 0$. The minimal polynomials of $\alpha, \alpha^3, \alpha^5$ are
Description of the Codes

\[
\therefore \phi_1(x) = 1 + x + x^4
\]

\[
\phi_3(x) = 1 + x + x^2 + x^3 + x^4
\]

\[
\phi_5(x) = 1 + x + x^5
\]

The double-error-correcting BCH code of length \( n = 2^4 - 1 = 15 \) is generated by \( g(x) = \text{LCM}\{\phi_1(x), \phi_3(x)\} \)

Since \( \phi_1(x) \) and \( \phi_3(x) \) are two distinct irreducible polynomials,

\[
g(x) = (1 + x + x^4)(1 + x + x^2 + x^3 + x^4)
\]

\[
= 1 + x^4 + x^6 + x^7 + x^8
\]

\[
\therefore (15, 7) \text{ cyclic code with } d_{\text{min}} \geq 5
\]

\[
\therefore W(g(x)) = 5 \quad \therefore d_{\text{min}} = 5
\]
Description of the Codes

- The triple-error-correcting BCH code of length 15 is generated by
  \[ g(x) = \text{LCM} \{ \phi_1(x), \phi_3(x), \phi_5(x) \} \]
  \[ = (1 + x + x^4)(1 + x + x^2 + x^3 + x^4)(1 + x + x^2) \]
  \[ = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10} \]
  It's a (15,5) cyclic code with \( d_{\text{min}} \geq 7 \). Since \( W(g(x)) = 7, d_{\text{min}} = 7 \).

- Let \( v(x) = v_0 + v_1x + \ldots + v_{n-1}x^{n-1} \) be a poly. with \( v_i \in GF(2) \)
  If \( v(x) \) has roots \( \alpha, \alpha^2, \ldots, \alpha^{2t} \), then \( v(x) \) is divisible by
  \( \phi_1(x), \phi_2(x), \ldots, \phi_{2t}(x) \). (Theorem 2.10)

- \( v(x) \) is a code poly. because \( v(x) | g(x) = \text{LCM}\{\phi_1(x), \phi_2(x), \ldots, \phi_{2t}(x)\} \)
Description of the Codes

We have a new definition for $t$-error-correcting BCH code:
A binary $n$-tuple $v = (v_0, v_1, v_2, ..., v_{n-1})$ is a code word if and only if the poly. $v(x) = v_0 + v_1x + ... + v_{n-1}x^{n-1}$ has $\alpha, \alpha^2, ..., \alpha^{2t}$ as roots i.e. $v(\alpha^i) = v_0 + v_1\alpha^i + v_2\alpha^{2i} + ... + v_{n-1}\alpha^{(n-1)i}$

\[
\begin{bmatrix}
1 \\
\alpha^i \\
\cdot \\
\cdot \\
\alpha^{(n-1)i}
\end{bmatrix}
\begin{bmatrix}
v_0 \\
v_1 \\
v_2 \\
\cdot \\
v_{n-1}
\end{bmatrix} = 0 \text{ for } 1 \leq i \leq 2t.
\]
Description of the Codes

Let

\[
H = \begin{bmatrix}
1 & \alpha & \ldots & \alpha^{n-1} \\
1 & (\alpha^2) & \ldots & (\alpha^2)^{n-1} \\
\vdots & \vdots & & \vdots \\
1 & (\alpha^{2t}) & \ldots & (\alpha^{2t})^{n-1}
\end{bmatrix}
\]

If \(v\) is a code word in the \(t\)-error-correcting BCH code, then

\[
v \cdot H^T = 0
\]

The code is the null space of the matrix \(H\) and \(H\) is the parity-check matrix of the code.
\( \alpha^j \) is a conjugate of \( \alpha^i \), then \( v(\alpha^j) = 0 \) iff \( v(\alpha^i) = 0 \) Thm. 2.7

\( j \)-th row of \( H \) can be omitted. As a result \( H \) can be reduced to the following form:

\[
H = \begin{bmatrix}
1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\
1 & \alpha^3 & (\alpha^3)^2 & \cdots & (\alpha^3)^{n-1} \\
1 & \alpha^{(2^t-1)} & (\alpha^{2^t-1})^2 & \cdots & (\alpha^{2^t-1})^{n-1}
\end{bmatrix}
\]

EX 6.2
double-error-correcting BCH code of length \( n = 2^4 - 1 = 15 \), \((15,7)\) code. Let \( \alpha \) be a primitive element in \( GF(2^4) \)
Description of the Codes

The parity-check matrix is

\[
H = \begin{bmatrix}
1 & \alpha & \alpha^2 & \ldots & \alpha^{14} \\
1 & \alpha^3 & \alpha^6 & \ldots & \alpha^{42}
\end{bmatrix}
\]

Using \(\alpha^{15} = 1\), and representing each entry of \(H\) by its 4-tuple,

\[
H = \begin{bmatrix}
1000100110101111 \\
010011010111100 \\
001001101011110 \\
000100110101111 \\
000100110101111 \\
100011000110001 \\
000110001100011 \\
001010010100101 \\
011110111101111
\end{bmatrix}
\]

(see p.149)
**FACT:**

The $t$-error-correcting BCH code indeed has $d_{\text{min}} \geq 2t + 1$

(pf): Suppose $\exists \; \mathbf{v} \neq \mathbf{0}$ such that $W(\mathbf{v}) = \delta \leq 2t$

Let $v_{j1}, v_{j2}, \ldots v_{j\delta}$ be the nonzero components of $\mathbf{v}$ (i.e. all ones)

$$0 = \mathbf{v}H^T = (v_{j1}v_{j2}\ldots v_{j\delta}) \begin{bmatrix} \alpha^{j1} (\alpha^2)^{j1} \cdots (\alpha^{2t})^{j1} \\ \alpha^{j2} (\alpha^2)^{j2} \cdots (\alpha^{2t})^{j2} \\ \vdots & \vdots \\ \alpha^{j\delta} (\alpha^2)^{j\delta} \cdots (\alpha^{2t})^{j\delta} \end{bmatrix}$$

$$\Rightarrow (1,1,\ldots,1) \begin{bmatrix} \alpha^{j1} (\alpha^{j1})^2 \cdots (\alpha^{j1})^{2t} \\ \alpha^{j2} (\alpha^{j2})^2 \cdots (\alpha^{j2})^{2t} \\ \vdots & \vdots \\ \alpha^{j\delta} (\alpha^{j\delta})^2 \cdots (\alpha^{j\delta})^{2t} \end{bmatrix} \equiv 0 \quad \cdots \otimes$$
Description of the Codes

\[
\begin{bmatrix}
\alpha^{j_1} (\alpha^{j_1})^2 \cdots (\alpha^{j_1})^\delta \\
\alpha^{j_2} (\alpha^{j_2})^2 \cdots (\alpha^{j_2})^\delta \\
\vdots & \vdots & \vdots \\
\alpha^{j_\delta} (\alpha^{j_\delta})^2 \cdots (\alpha^{j_\delta})^\delta
\end{bmatrix}
= 0
\]

\(A,\) a \(\delta \times \delta\) square matrix \((\delta \leq 2t)\)

\[\Rightarrow |A| = 0\]

\[\Rightarrow \alpha^{(j_1+j_2+\ldots+j_\delta)} \begin{vmatrix}
1\alpha^{j_1} \cdots \alpha^{(\delta-1)j_1} \\
1\alpha^{j_2} \cdots \alpha^{(\delta-1)j_2} \\
\vdots & \vdots & \vdots \\
1\alpha^{j_\delta} \cdots \alpha^{(\delta-1)j_\delta}
\end{vmatrix} = 0 \ldots \otimes \otimes\]
Description of the Codes

- The determinant in the equality above is a **Vandermonde determinant** which is nonzero. The product on the left-hand side of $\otimes \otimes$ cannot be zero. This is a contradiction and hence our assumption that there exists a nonzero code vector $v$ of $W(v) = \delta \leq 2t \implies d_{\text{min}} \geq 2t + 1$ is invalid.

- $2t+1$ is the **designed distance** of the $t$-error-correcting BCH code. The true minimum distance of a BCH code may or may not be equal to its designed distance.

- Binary BCH code with length $n \neq 2^m - 1$ can be constructed in the same manner as for the case $n = 2^m - 1$.
  
  Let $\beta$ be an element of order $n$ in $GF(2^m)$, $n \mid 2^m - 1$ and $g(x)$ be the binary polynomial of minimum degree that has $\beta, \beta^2, \ldots, \beta^{2t}$ as roots.
Let $\psi_1(x), \psi_2(x), \ldots, \psi_{2t}(x)$ be the minimal poly. of $\beta, \beta^2, \ldots, \beta^{2t}$ respectively then

$$g(x) = \text{LCM}\{\psi_1(x), \psi_2(x), \ldots, \psi_{2t}(x)\}$$

$\therefore \beta^n = 1, \therefore \beta, \beta^2, \ldots, \beta^{2t}$ are roots of $x^n + 1$

$$\Rightarrow g(x) \mid (x^n + 1)$$

We see that $g(x)$ is a factor of $X^n + 1$.

- The cyclic code generated by $g(x)$ is a $t$-error-correcting BCH code of length $n$.
- The number of parity-check digits $\leq mt$
- $d_{\min} \geq 2t + 1$.

If $\beta$ is not a primitive element of $\text{GF}(2^m)$, the code is called a nonprimitive BCH code.
Description of the Codes

- General definition of binary BCH codes.
  \[ \beta \in GF(2^m), \ell_0 \] be any nonnegative integer and consider
  \[ \beta^{\ell_0}, \beta^{\ell_0+1},..., \beta^{\ell_0+d_0-2}. \]
  For \( 0 \leq i \leq d_0 - 1 \), let \( \psi_i(x), n_i \) be the
  minimal poly. and order of \( \beta^{\ell_0+i} \), respectively.

  \[ g(x) = \text{LCM}\{\psi_0(x), \psi_1(x),..., \psi_{d_0-2}(x)\} \]

  and the length of the code is
  \[ n = \text{LCM}\{n_0, n_1,...n_{d_0-2}\} \]

  Note that: \( d_{\text{min}} \geq d_0 \)
  parity-check digits \( \leq m(d_0 - 1) \)
  is capable of correcting \([ (d_0 - 1) / 2 ] \) or fewer errors
Description of the Codes

If we let $l_0 = 1$, $d_0 = 2t+1$ and $\beta$ be a primitive element of $\text{GF}(2^m)$, the code becomes a $t$-error-correcting primitive BCH code of length $2^m - 1$.

If we let $l_0 = 1$, $d_0 = 2t+1$ and $\beta$ be not a primitive element of $\text{GF}(2^m)$, the code is a nonprimitive $t$-error-correcting BCH code of length $n$, which is the order of $\beta$.

For a BCH code with designed distance $d_0$, we require $g(x)$ has $d_0-1$ consecutive powers of a field element $\beta$ as roots. This guarantees that the code has $d_{\min} \geq d_0$. This lower bound on the minimum distance is called the BCH bound.

In the rest of this chapter, we consider only the primitive BCH codes.
Decoding of the BCH Codes

National Sun Yat-sen University
Decoding of the BCH Codes

Suppose that a code word \( v(x) = v_0 + v_1 x + v_2 x^2 + \ldots + v_{n-1} x^{n-1} \)
is transmitted and the transmission errors result:
\[
\mathbf{r}(x) = r_0 + r_1 x + r_2 x^2 + \ldots + r_{n-1} x^{n-1}
\]
Let \( e(x) \) be the error pattern. Then
\[
\mathbf{r}(x) = \mathbf{v}(x) + \mathbf{e}(x)
\]
For decoding, remember
\[
\mathbf{H} = \begin{bmatrix}
1 & \alpha & \ldots & \alpha^{n-1} \\
1 & (\alpha^2) & \ldots & (\alpha^2)^{n-1} \\
\vdots & \ddots & \ddots & \vdots \\
1 & (\alpha^{2t}) & \ldots & (\alpha^{2t})^{n-1}
\end{bmatrix}
\]
Decoding of the BCH Codes

The syndrome is 2t-tuple,

\[ S = (S_1, S_2, \ldots, S_{2t}) = r \cdot H^T \]

Let \( s_i = r(\alpha^i) = r_0 + r_1\alpha^i + \ldots + r_{n-1}(\alpha^i)^{n-1} = 0 \) for \( 1 \leq i \leq 2t \)

\( s_i \) can be evaluated by \( b_i(x) = R_{\phi_i(x)}[r(x)] \)

\( \phi_i(x) \) is the minimal poly. of \( \alpha^i \)

\[ r(x) = a_i(x)\phi_i(x) + b_i(x) \]

\[ s_i = r(\alpha^i) = b_i(\alpha^i) \]
Decoding of the BCH Codes

EX6.4

Consider the double-error-correcting (15, 7) BCH code given in (from Ex6.1). If \( r = (100000001000000) \) \( \Rightarrow r(x) = 1 + x^8 \)

\[
\phi_1(x) = \phi_2(x) = \phi_4(x) = 1 + x + x^4 \quad \therefore s_1 = b_1(\alpha) = \alpha^2, s_2 = b_1(\alpha^2) = \alpha^4
\]

\[
\phi_3(x) = 1 + x + x^2 + x^3 + x^4 \quad s_3 = b_3(\alpha^3) = 1 + \alpha^9
\]

\[
b_1(x) = R_{\phi_1(x)}[r(x)] = x^2 \quad = 1 + \alpha + \alpha^3 = \alpha^7
\]

\[
b_3(x) = R_{\phi_3(x)}[r(x)] = 1 + x^3 \quad s_4 = b_1(\alpha^4) = \alpha^8
\]

\[
\therefore s = (\alpha^2, \alpha^4, \alpha^7, \alpha^8)
\]
Decoding of the BCH Codes

\[ v(\alpha^i) = 0 \quad \text{for} \quad 1 \leq i \leq 2t \rightarrow s_i = r(\alpha^i) = e(\alpha^i) \]

Suppose \( e(x) = x^{j_1} + x^{j_2} + \ldots + x^{j_v} \quad 0 \leq j_1 < j_2 < \ldots j_v < n \)

\[ \Rightarrow s_1 = \alpha^{j_1} + \alpha^{j_2} + \ldots + \alpha^{j_v} \]

\[ s_2 = (\alpha^{j_1})^2 + (\alpha^{j_2})^2 + \ldots + (\alpha^{j_v})^2 \]

\[ \vdots \]

\[ s_{2t} = (\alpha^{j_1})^{2t} + \ldots + (\alpha^{j_v})^{2t} \]

Where \( \alpha^{j_1}, \alpha^{j_2}, \ldots \alpha^{j_v} \) are unknown.

Any method for solving these equations is a decoding algorithm for the BCH codes.
Decoding of the BCH Codes

Once $\alpha^{j_1}, \alpha^{j_2}, \ldots, \alpha^{j_v}$ have been found, the powers $j_1, j_2, \ldots, j_v$ tell us the error locations in $e(x)$.

If the number of errors in $e(x)$ is $t$ or less, the solution that yields an error pattern with the **smallest number of errors** is the right solution.

For convenience, let $\beta_\ell = \alpha^{j_\ell}$, $1 \leq \ell \leq v$ be the **error location numbers**.

\[
\begin{align*}
    s_1 &= \beta_1 + \beta_2 + \ldots + \beta_v \quad \text{Power-sum} \\
    s_2 &= \beta_1^2 + \beta_2^2 + \ldots + \beta_v^2 \\
    \vdots &\quad \text{Symmetric function} \\
    s_{2t} &= \beta_1^{2t} + \beta_2^{2t} + \ldots + \beta_v^{2t}
\end{align*}
\]
Decoding of the BCH Codes

Define

\[\sigma(x) = (1 + \beta_1 x)(1 + \beta_2 x)\ldots(1 + \beta_v x)\]

\[= \sigma_0 + \sigma_1 x + \sigma_2 x^2 + \ldots + \sigma_v x^v\]

: error-location poly.

The roots of \(\sigma(x)\) are \(\beta_1^{-1}, \beta_2^{-1}, \ldots \beta_v^{-1}\), which are the inverses of the error location numbers.

\[\sigma_0 = 1\]
\[\sigma_1 = \beta_1 + \beta_2 + \ldots \beta_v\]
\[\sigma_2 = \beta_1 \beta_2 + \beta_2 \beta_3 + \ldots + \beta_{v-1} \beta_v\]
\[\vdots\]
\[\sigma_v = \beta_1 \beta_2 \ldots \beta_v\]
Decoding of the BCH Codes

These coefficients are known as **elementary symmetric functions**. The \( \sigma_i \)'s are related to \( s_j \)'s by Newton's identities:

\[
\begin{align*}
\sigma_1 &= s_1 = 0 \\
\sigma_1 s_1 + 2\sigma_2 &= s_2 = 0 \\
\sigma_1 s_2 + \sigma_2 s_1 + 3\sigma_3 &= s_3 = 0 \\
&\quad \vdots \\
\sigma_1 s_{v-1} + \ldots + \sigma_{v-1} s_1 + v\sigma_v &= s_v = 0 \\
\sigma_1 s_v + \ldots + \sigma_{v-1} s_2 + \sigma_v s_1 &= s_{v+1} = 0
\end{align*}
\]

Note that, for binary case, \( 1 + 1 = 2 = 0 \). We have

\[
i\sigma_i = \begin{cases} 
\sigma_i & \text{for odd } i \\
0 & \text{for even } i
\end{cases}
\]
Decoding of the BCH Codes

- The equations may have many solutions.
- We want to find the solution that yields a $\sigma(X)$ of minimal degree. This $\sigma(X)$ would produce an error pattern with minimum number of errors.

Decoding Procedure

1. Compute $s = (s_1, s_2, \ldots, s_t)$ from $r(x)$
2. Determine $\sigma(x)$ from $s$
3. Determine the error-location number $\beta_1, \beta_2, \ldots, \beta_v$ by finding the roots of $\sigma(x)$
   and correct the errors in $r(x)$
Decoding of the BCH Codes

Iterative Algorithm for finding $\sigma(x)$ (Berlekamps iterative algorithm)

1. $i = 1$
   
   $\sigma^{(1)} = 1 + s_1 x$

   \[ \sigma = 1 + s_1 x \]

2. $i = 2t$

   \[ \sigma(x) = \sigma^{(2t)}(x) \]

3. Does the coefficients of $\sigma^{(i)}$ satisfy the $(i+1)$th Newton's identity?

   - Yes: $\sigma^{(i+1)} = \sigma^{(i)}$
   - No: $\sigma^{(i+1)} = \sigma^{(i)} + a$

4. $i = i + 1$

   \[ i = i + 1 \]
Decoding of the BCH Codes

How to add a correction term to $\sigma^{(i)}$?

Let $\sigma^{(\mu)} = 1 + \sigma_1^{(\mu)} x + \sigma_2^{(\mu)} x^2 + \ldots + \sigma_{\ell\mu}^{(\mu)} x^{\ell\mu}$ be the minimum-degree poly. determined at the $\mu$th step.

To determine $\sigma^{(\mu+1)}(x) \Rightarrow$ compute $\mu$th discrepancy

$$d_\mu = s_{\mu+1}^{(\mu)} + \sigma_1^{(\mu)} s_\mu^{(\mu)} + \sigma_2^{(\mu)} s_{\mu-1}^{(\mu)} + \ldots + \sigma_{\ell\mu}^{(\mu)} s_{\mu+1-\ell\mu}^{(\mu)}$$

If $d_\mu = 0 \Rightarrow \sigma^{(\mu+1)}(x) = \sigma^{(\mu)}(x)$

If $d_\mu \neq 0$, go back to the steps prior to the $\mu$th step and determine $\sigma^{(\rho)}(x)$ s.t. $\rho$th discrepancy $d_\rho \neq 0$, and $\rho - \ell\rho$ has the largest value. $(\ell\rho = \text{deg } [\sigma^{(\rho)}(x)])$. Then

$$\sigma^{(\mu+1)}(x) = \sigma^{(\mu)}(x) + d_\mu d_\rho^{-1} x^{(\mu-\rho)} \sigma^{(\rho)}(x)$$
Decoding of the BCH Codes

is the minimum-degree poly. whose coefficients satisfy the first $\mu + 1$ Newton’s identities

To carry out the iteration of finding $\sigma(X)$, we begin with the following table:

<table>
<thead>
<tr>
<th>$\mu$</th>
<th>$\sigma^{(\mu)}(x)$</th>
<th>$d_\mu$</th>
<th>$\ell_\mu$</th>
<th>$\mu - \ell_\mu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$S_i$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$2t$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---
Decoding of the BCH Codes

- $l_\mu$ is the degree of $\sigma^{(\mu)}(X)$.
- If $d_\mu = 0$, then $\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X)$ and $l_{\mu+1} = l_{\mu}$.
- If $d_\mu \neq 0$, find another row $\rho$ prior to the $\mu$th row such that $d_\rho \neq 0$ and the number $\rho - l_\rho$ in the last column of the table has the largest value. Then $\sigma^{(\mu+1)}(X)$ is given by

$$\sigma^{(\mu+1)}(x) = \sigma^{(\mu)}(x) + d_\mu d_\rho^{-1} x^{(\mu-\rho)} \sigma^{(\rho)}(x)$$

and $l_{\mu+1} = \max(l_\mu, l_\rho + \mu - \rho)$.
- In either case, $d_{\mu+1} = s_{\mu+2} + \sigma_1^{(\mu+1)} s_{\mu+1} + \ldots + \sigma_\ell^{(\mu+1)} s_{\mu+2-\ell_{\mu+1}}$
- The polynomial $\sigma^{(2\ell)}(X)$ in the last row should be the required $\sigma(X)$. 
Decoding of the BCH Codes

Ex 6.5
Consider (15.5) triple-error-correcting BCH codes given in ex 6.1

\[ p(x) = 1 + x + x^4 \]

\[ \nu = 0 \rightarrow \oplus \rightarrow r = x^3 + x^5 + x^{12} \]

\[ \phi_1(x) = \phi_2(x) = \phi_4(x) = 1 + x + x^4 \]

\[ \phi_3(x) = \phi_6(x) = 1 + x + x^2 + x^3 + x^4 \]

\[ \phi_5(x) = 1 + x + x^2 \]
Decoding of the BCH Codes

\[ b_1(x) = R_{\phi_1(x)}[r(x)] = 1 \]
\[ b_3(x) = R_{\phi_3(x)}[r(x)] = 1 + x^2 + x^3 \]
\[ b_5(x) = R_{\phi_5(x)}[r(x)] = x^2 \]
\[ s_1 = s_2 = s_4 = 1 \]
\[ s_3 = 1 + \alpha^6 + \alpha^9 = \alpha^{10} \]
\[ s_6 = 1 + \alpha^{12} + \alpha^{18} = \alpha^5 \]
\[ s_5 = \alpha^{10} \]

\[ \therefore \underline{s} = (1, 1, \alpha^{10}, 1, \alpha^{10}, \alpha^5) \]

\[ \sigma^{(\mu+1)}(x) = \sigma^{(\mu)}(x) + d_{\mu} d_{\rho}^{-1} x^{(\mu-\rho)} \sigma^{(\rho)}(x) \]
Decoding of the BCH Codes

- $S_1 = 1$
- $d_0 = 1 \neq 0$
- $\rho = -1$
- $\sigma^{(1)}(X) = \sigma^{(0)}(X) + d_0 d_{-1}^{-1} X^{0+1} \sigma^{(-1)}(X) = 1 + 1 \cdot 1 \cdot X \cdot 1 = 1 + X$
- $l_1 = \max(l_0, l_{-1} + \mu - \rho) = \max(0, 0 - 0 + 1) = 1$
- $\mu - l_{\mu} = 1 - l_1 = 1 - 1 = 0$
- $d_1 = S_2 + \sigma_1^{(1)} S_1 = 1 + 1 \cdot 1 = 0$

- $\sigma^{(2)}(X) = \sigma^{(1)}(X) = 1 + X$
- $l_2 = l_1 = 1$
- $\mu - l_{\mu} = 2 - l_2 = 2 - 1 = 1$
- $d_2 = S_3 + \sigma_1^{(2)} S_2 + \sigma_2^{(2)} S_1 = \alpha^{10} + 1 \cdot 1 + 0 \cdot 1 = (1 + \alpha + \alpha^2) + 1 = \alpha^5$
Decoding of the BCH Codes

- \( d_2 = \alpha^5 \neq 0 \)
- \( \rho = 0 \)
- \( \sigma^{(3)}(X) = \sigma^{(2)}(X) + d_2 d_0^{-1} X^{(2-0)} \sigma^{(0)}(X) = 1 + X + \alpha^5 \cdot 1 \cdot X^2 \cdot 1 = 1 + X + \alpha^5 X^2 \)
- \( l_3 = \max(l_2, l_0 + \mu - \rho) = \max(1, 0 + 2 - 0) = 2 \)
- \( \mu - l_\mu = 3 - l_3 = 3 - 2 = 1 \)
- \( d_3 = S_4 + \sigma_1^{(3)} S_3 + \sigma_2^{(3)} S_2 + \sigma_3^{(3)} S_1 = 1 + 1 \cdot \alpha^{10} + \alpha^5 \cdot 1 + 0 \cdot 1 = 1 + (1 + \alpha + \alpha^2) + (\alpha + \alpha^2) = 0 \)
Decoding of the BCH Codes

<table>
<thead>
<tr>
<th>μ</th>
<th>$\sigma^{(\mu)}(x)$</th>
<th>$d_\mu$</th>
<th>$\ell_\mu$</th>
<th>$\mu - \ell_\mu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$1 + x$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$1 + x$</td>
<td>$\alpha^5$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>$1 + x + \alpha^5x^2$</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$1 + x + \alpha^5x^2$</td>
<td>$\alpha^{10}$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>$1 + x + \alpha^5x^3$</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>$1 + x + \alpha^5x^3$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

∴ $\sigma(x) = \sigma^{(6)}(x) = 1 + x + \alpha^5x^3$

\[= (1 + \alpha^{-3}x)(1 + \alpha^{-10}x)(1 + \alpha^{-12}x)\]

∴ $x = \alpha^3, \alpha^{10}, \alpha^{12}$
Decoding of the BCH Codes

error location numbers
\[ \beta_1 = \alpha^{-3} = \alpha^{12}, \beta_2 = \alpha^{-10} = \alpha^5, \beta_3 = \alpha^{-12} = \alpha^3 \]
\[ \therefore e(x) = x^3 + x^5 + x^{12} \]
\[ \therefore r(x) = r(x) + e(x) = 0 \]

✠ If the number of errors in the received polynomial \( r(X) \) is less than the designed error-correcting capability \( t \) of the code, it is not necessary to carry out the \( 2t \) steps of iteration to find the error-location polynomial \( \sigma(X) \).

✠ It has been shown that if \( d_\mu \) and the discrepancies at the next \( t-l_\mu -1 \) steps are all zeros (i.e. successive \( t-l_\mu \) zeros), \( \sigma^{(\mu)}(X) \) is the error-location polynomial.

✠ If \( v(\leq t) \) errors occur, only \( v+t \) steps of iteration are needed.

✠ The iterative algorithm described above not only applies to binary BCH codes but also nonbinary BCH codes.
### Simplified Algorithm for finding $\sigma(x)$

For a binary BCH code, it is only required to fill out a table with $t$ empty rows. Such a table is presented below.

<table>
<thead>
<tr>
<th>$\mu$</th>
<th>$\sigma^{(\mu)}(x)$</th>
<th>$d_\mu$</th>
<th>$I_\mu$</th>
<th>$2\mu - I_\mu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1/2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$S_1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Decoding of the BCH Codes

1. If \( d_{\mu} = 0 \), then \( \sigma^{(\mu+1)}(x) = \sigma^{(\mu)}(x) \)

2. If \( d_{\mu} \neq 0 \), find another row \( \rho \) preceding the \( \mu \)th row, s.t. \( 2\rho - \ell_{\rho} \) is as large as possible and \( d_{\rho} \neq 0 \)

Then \( \sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X) + d_{\mu} d_{\rho}^{-1} X^{2(\mu-\rho)} \sigma^{(\rho)}(X) \)

note that \( d_{\mu+1} = s_{2\mu+3}^{(\mu+1)} + \sigma_{1}^{(\mu+1)} s_{2\mu+2}^{(\mu+1)} + \sigma_{2}^{(\mu+1)} s_{2\mu+1}^{(\mu+1)} \)

\[ + \ldots + \sigma_{\ell_{\mu+1}}^{(\mu+1)} s_{2\mu+3-\ell_{\mu+1}}^{(\mu+1)} \]

\( \ell_{\mu+1} = \deg [\sigma^{(\mu+1)}(x)] \)
Decoding of the BCH Codes

- The polynomial $\sigma^{(t)}(X)$ in the last row should be the required $\sigma(X)$. If it has degree greater than $t$, there were more than $t$ errors, and generally it is not possible to locate them.
- The computation required in this simplified algorithm is one-half of the computation required in the general algorithm.
- The simplified algorithm applies only to binary BCH codes.
- If the number of errors in the received polynomial $r(X)$ is less than the designed error-correcting capability $t$ of the code, it is not necessary to carry out the $t$ steps of iteration to find the error-location polynomial $\sigma(X)$ for a $t$-error-correcting binary BCH code.
Decoding of the BCH Codes

**Remarks:**
- If \( v \leq t \) errors occur, only \( \lceil (t + v) / 2 \rceil \) steps needed.
- If, for some \( \mu \), \( d_\mu \) and the discrepancies at the next \( (t - l_\mu - 1)/2 \) steps are zero, then \( \sigma(X) \) is the error-location poly.

**EX6.6**
The simplified table for finding \( \sigma(x) \) for the code in ex6.5 is given below. Thus, \( \sigma(x) = \sigma^{(3)}(x) = 1 + x + \alpha^5 x^3 \).

<table>
<thead>
<tr>
<th>( \mu )</th>
<th>( \sigma^{(\mu)}(x) )</th>
<th>( d_\mu )</th>
<th>( I_\mu )</th>
<th>( 2 \mu - I_\mu )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1/2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>( S_1 = 1 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>( 1 + S_1 x = 1 + x )</td>
<td>( S_3 + S_2 S_1 = \alpha^5 )</td>
<td>1</td>
<td>1 (take ( \rho = -1/2 ))</td>
</tr>
<tr>
<td>2</td>
<td>( 1 + x + \alpha^5 x^2 )</td>
<td>( \alpha^{10} )</td>
<td>2</td>
<td>2 (take ( \rho = 0 ))</td>
</tr>
<tr>
<td>1</td>
<td>( 1 + x + \alpha^5 x^3 )</td>
<td>----</td>
<td>3</td>
<td>3 (take ( \rho = 1 ))</td>
</tr>
</tbody>
</table>
Decoding of the BCH Codes

Finding the Error-Location Numbers and Error Correction.

Consider ex6.6. The error-location poly. has been found to be

\[ \sigma(x) = 1 + x + \alpha^5 x^3 \]

By substituting \( 1, \alpha, \alpha^2, \ldots, \alpha^{14} \) into it, we find that \( \alpha^3, \alpha^{10}, \alpha^{12} \) are the roots of \( \sigma(x) \). Therefore, the error location numbers are \( \alpha^{12}, \alpha^5, \alpha^3 \Rightarrow e(x) = x^3 + x^5 + x^{12} \)

Chien’s procedure: The received vector

\[ r(x) = r_0 + r_1 x + r_2 x^2 + \ldots + r_{n-1} x^{n-1} \]

is decoded on a bit-by-bit basis. The high-order bits are decoded first. To decode \( r_{n-1} \), the decoder test whether \( \alpha^{n-1} \) is an error-location number; this is equivalent to test whether its inverse \( \alpha \) is a root of \( \sigma(x) \). If \( \alpha \) is a root, then

\[ 1 + \sigma_1 \alpha + \sigma_2 \alpha^2 + \ldots + \sigma_v \alpha^v = 0 \]
To decode $r_{n-l}$, the decoder forms $\sigma_1 \alpha^l, \sigma_2 \alpha^{2l}, ..., \sigma_v \alpha^{vl}$ and tests the sum

$$1 + \sigma_1 \alpha^l + \sigma_2 \alpha^{2l} + ... + \sigma_v \alpha^{vl}$$

If the sum is zero, then $\alpha^{n-l}$ is an error-location number and $r_{n-l}$ is an erroneous digit; otherwise, $r_{n-l}$ is a correct digit.

Cyclic error location search unit
Decoding of the BCH Codes

The $t$ $\sigma$-registers are initially stored with $\sigma_1, \sigma_2, \ldots, \sigma_t$ calculated in step 2 of the decoding (for $\sigma_1 = \sigma_2 = \ldots = \sigma_v = 0$ for $v < t$). Immediately before $r_{n-1}$ is read out of the buffer, the $t$ multipliers are pulsed once. The multiplications are performed and $\sigma_1\alpha, \sigma_2\alpha^2, \ldots, \sigma_v\alpha^v$ are stored in the $\sigma$-registers. The output of the logic circuit $A$ is 1 if and only if the sum $1 + \sigma_1\alpha + \sigma_2\alpha^2 + \ldots + \sigma_v\alpha^v = 0$; otherwise, the output of $A$ is 0. The digit $r_{n-1}$ is read out of the buffer and corrected by the output of $A$. Having decoded $r_{n-1}$, the $t$ multipliers are pulsed again. Now $\sigma_1\alpha^2, \sigma_2\alpha^4, \ldots, \sigma_v\alpha^{2v}$ are stored in the $\sigma$-registers. The sum

$$1 + \sigma_1\alpha^2 + \sigma_2\alpha^4 + \ldots + \sigma_v\alpha^{2v}$$

is tested for 0. The digit $r_{n-2}$ is read out of the buffer and corrected in the same manner as $r_{n-1}$ is corrected.
Implementation of Galois Field Arithmetic
Galois field adder (GF(2^4)) given by Table 2.8

By adding their vector representations with modulo-2 addition.
For multiplication, we first consider multiplying a field element by a fixed element from the same field. Suppose that we want to multiply a field element $\beta$ in the $\mathbb{GF}(2^4)$ by the primitive element $\alpha$ whose minimal polynomial is $\phi(x) = 1 + x + x^4$.

$$\beta = b_0 + b_1\alpha + b_2\alpha^2 + b_3\alpha^3$$

$$\alpha\beta = b_3 + (b_0 + b_3)\alpha + b_1\alpha^2 + b_2\alpha^3 \iff \alpha^4 = 1 + \alpha$$

Ex1:

Let $\beta = \alpha^7 = 1 + \alpha + \alpha^3$. The vector representation of $\beta$ is $(1\ 1\ 0\ 1)$. Load it into the register. After the register is pulsed, the new content in it will be $(1\ 0\ 1\ 0)$. 
the above circuit can be used to generate all the nonzero elements of \( GF(2^4) \) by load (1000) first into the register

\* Ex2: as another example, multiply \( \beta \) by \( \alpha^3 \):

\[
\beta = b_0 + b_1\alpha + b_2\alpha^2 + b_3\alpha^3
\]

\[
\alpha^3 \beta = b_0\alpha^3 + b_1\alpha^4 + b_2\alpha^5 + b_3\alpha^5
\]

\[
= b_0\alpha^3 + b_1(1+\alpha) + b_2(\alpha + \alpha^2)
\]

\[
+ b_3(\alpha^2 + \alpha^3)
\]

\[
= b_1 + (b_1 + b_2)\alpha + (b_2 + b_3)\alpha^2 + (b_0 + b_3)\alpha^3
\]
Ex3:

multiplying two arbitrary elements

\[ \beta = b_0 + b_1 \alpha + b_2 \alpha^2 + b_3 \alpha^3 \]
\[ r = c_0 + c_1 \alpha + c_2 \alpha^2 + c_3 \alpha^3 \]
\[ \beta r = c_0 \beta + c_1 \beta \alpha + c_2 \beta \alpha^2 + c_3 \beta \alpha^3 \]
\[ = (((c_3 \beta) \alpha + c_2 \beta) \alpha + c_1 \beta) \alpha + c_0 \beta \]

Steps:

1. multiply \( c_3 \beta \) by \( \alpha \) and add the product to \( c_2 \beta \)
2. multiply \( (c_3 \beta) \alpha + c_2 \beta \) by \( \alpha \) and add the product to \( c_1 \beta \)
3. multiply \( ((c_3 \beta) \alpha + c_2 \beta) \alpha + c_1 \beta \) by \( \alpha \) and add the product to \( c_0 \beta \)
The circuit in ex2 can be modified to carry out the computation given by ex3.
Step 1: Register A are empty

\[ B \leftarrow (b_0, b_1, b_2, b_3) \quad C \leftarrow (c_0, c_1, c_2, c_3) \]

Step 2: Registers A, C are shifted four times.

At the end of first shift, \( A \leftarrow ((c_3 b_0, c_3 b_1, c_3 b_2, c_3 b_3) \), the vector representation of \( c_3 \beta \)
At the end of second shift, \( A \leftarrow (((c_3 \beta) \alpha + c_2 \beta) \)
At the end of third shift, \( A \leftarrow (((c_3 \beta) \alpha + c_2 \beta) \alpha + c_1 \beta) \)
At the end of fourth shift, \( A \leftarrow \beta \gamma \)
If we express

\[ \beta \gamma = (((c_0 \beta) + c_1 \beta \alpha) + c_2 \beta \alpha^2) + c_3 \beta \alpha^3 \]

then we obtain a different multiplication circuit.
Implementation of Galois Field Arithmetic

Multiplication can be implemented by combinational logic circuit with $2m$ inputs and $m$ outputs.

- advantage $\rightarrow$ speed
- disadvantage $\rightarrow$ $m > 7$, prohibitively complex and costly.

Programmed in a general-purpose computer will require roughly $5m$ instruction executions.

How to compute syndrome $s_i = r(\alpha^i)$?

It can be done with a circuit for multiplying a field element by $\alpha^i$. Suppose that we want to compute

$$r(\alpha) = r_0 + r_1\alpha + r_2\alpha^2 + \ldots + r_{14}\alpha^{14}$$

$$= (\ldots(((r_{14})\alpha + r_{13})\alpha + r_{12})\alpha + \ldots)\alpha + r_0$$

where $\alpha$ is a primitive element in GF($2^4$)
The computation of \( r(\alpha) \) can be accomplished by adding an input to the circuit:

At the end of first shift \((r_{14}, 0, 0, 0)\)

At the end of 2nd shift \(r_{14}\alpha + r_{13}\)

At the end of 3rd shift \((r_{14}\alpha + r_{13})\alpha + r_{12}\)
We can compute \( r(\alpha^3) \) by adding an input to the circuit for multiplying by \( \alpha^3 \) of figure in Ex2.

There is another way of computing \( r(\alpha^i) \). Let \( \phi_i(x) \) be the minimal poly. of \( \alpha^i \). Let \( b(x) \) be the remainder resulting from dividing \( r(x) \) by \( \phi_i(x) \). Then

\[
r(\alpha^i) = b(\alpha^i)
\]
Thus, computing $r(\alpha^i)$ is equivalent to computing $b(\alpha^i)$. Again we consider computation over $GF(2^4)$. Suppose that we want to compute $r(\alpha^3)$. The minimal poly. of $\alpha^3$ is

$$\phi_3(x) = 1 + x + x^2 + x^3 + x^4$$

Then:

$$b(x) = b_0 + b_1x + b_2x^2 + b_3x^3$$

Thus,

$$b(\alpha^3)$$

$$= b_0 + b_1\alpha^3 + b_2\alpha^6 + b_3\alpha^9$$

$$= b_0 + b_1\alpha^3 + b_2(\alpha + \alpha^3) + b_3(\alpha + \alpha^3)$$

$$= b_0 + b_3\alpha + b_2\alpha^2 + (b_1 + b_2 + b_3)\alpha^3$$
Implementation of Galois Field Arithmetic

Such a circuit is shown below
Since $\alpha^6$ is a conjugate of $\alpha^3$, it has the same poly. as $\alpha^3$

\[
b(\alpha^6) = b_0 + b_1\alpha^6 + b_2\alpha^{12} + b_3\alpha^{18}
\]

\[
= b_0 + b_1(\alpha^2 + \alpha^3) + b_2(1 + \alpha + \alpha^2 + \alpha^3) + b_3\alpha^3
\]

\[
= (b_0 + b_2) + b_2\alpha + (b_1 + b_2)\alpha^2 + (b_1 + b_2 + b_3)\alpha^3
\]

Its circuit is shown in the next page
Implementation of Galois Field Arithmetic
Syndrome Computation

The first step in decoding $t$-error-correction BCH code is to compute the $2t$ syndrome component $S_1, S_2, \ldots, S_{2t}$.

Software: $S_i = r(\alpha^i) = r_{n-1}(\alpha^i)^{n-1} + r_{n-2}(\alpha^i)^{n-2} + \ldots + r_1\alpha^i + r_0$

$= \ldots ((r_{n-1}\alpha^i + r_{n-2})\alpha^i + r_{n-3})\alpha^i + \ldots + r_1\alpha^i + r_0$

$(n-1)$ addition and $(n-1)$ multiplication

For binary $S_{2i} = S_i^2$ : $2t$ syndrome components can be computed with $(n-1)t$ additions and $nt = (n-1)t + t$ multiplications.
For hardware implementation, a variety of types of circuits can be adopted, e.g. Figure 6.7 and 6.8, or Figure 6.10.

Alternatively, as shown in Figure 6.11 (the following page), at most $t$ feedback shift registers, each consisting of at most $m$ stages, are needed to form the $2t$ syndrome components because the generator polynomial is a product of at most $t$ minimal polynomials.
Syndrome computation circuit for the double-error-correcting (15,7) BCH code
As soon as the entire $r(x)$ has entered the decoder, the $2t$ syndrome components are formed.

It takes $n$ clock cycles to complete the computation.

Finding the Error-Location Polynomial $\sigma(X)$
Software computation requires somewhat less than $t$ additions and $t$ multiplications to compute each $\sigma^{(\mu)}(X)$ and each $d_\mu$, and since there are $t$ of each, the total is roughly $2t^2$ additions and $2t^2$ multiplications.
Implementation of Error Correction

hardware: requires the same total, and the speed would depend on how much is done in parallel.

Computations of Error-location Numbers and Error Correction
Worst case: requires substituting $n$ field elements into an error-location poly. $\sigma(x)$ of degree $t$ to determine roots.
Software: requiring $nt$ multiplications and $nt$ additions.
Hardware: using Chien’s searching circuit. See Figure 6.1.
   It requires $t$ multipliers for multiplying by $\alpha, \alpha^2, \ldots, \alpha^t$ respectively.
At the end of the \( \ell th \) shift, If

\[
\text{sum} = 1 + \sigma_1 \alpha^l + \sigma_2 \alpha^{2^l} + \ldots + \sigma_t \alpha^{t^l}
\]

\( \text{sum}=0, \ \alpha^{n-\ell} \) is an error-location number. The sum can be formed by using \( t m \)-input modulo-2 adders. A \( m \)-input OR gate is used to test whether the sum is zero. It takes \( n \) clock cycles to complete this step. If we only to correct the massage digits, only \( k \) clock cycles are needed. A Chien’s searching circuit for (15, 7, 2) BCH code is shown next slide.
Implementation of Error Correction
Nonbinary BCH Codes and Reed-Solomon Codes
Nonbinary BCH Codes and Reed-Solomon Codes

- In addition to the binary codes, there are nonbinary codes.
- If $p$ is a prime number and $q$ is any power of $p$ ($q=p^m$), there are codes with symbols from the Galois field $\text{GF}(q)$. These codes are called $q$-ary codes.
- An $(n, k)$ linear code with symbols from $\text{GF}(q)$ is a $k$-dimensional subspace of the vector space of all $n$-tuples over $\text{GF}(q)$.
- A $q$-ary $(n, k)$ cyclic code is generated by a polynomial of degree $(n-k)$ with coefficients from $\text{GF}(q)$, which is a factor of $X^n-1$.
- Encoding and decoding of $q$-ary codes are similar to that of binary codes.
- For any choice of positive integers $s$ and $t$, there exists a $q$-ary BCH code of length $n=q^s-1$, which can correct any combinational of $t$ or fewer errors and requires no more than $2st$ parity-check digits.
Nonbinary BCH Codes and Reed-Solomon Codes

Let $\alpha$ be a primitive element in the Galois field $\text{GF}(q^s)$.

The generator polynomial $g(X)$ of a $t$-error-correcting $q$-ary BCH is the polynomial of lowest degree with coefficients from $\text{GF}(q)$ for which $\alpha, \alpha^2, \ldots, \alpha^{2t}$ are roots.

Let $\Phi_i(X)$ be the minimal polynomial of $\alpha^i$. Then

$$g(X) = \text{LCM}\{ \Phi_1(X), \Phi_2(X), \ldots, \Phi_{2t}(X) \}$$

The degree of each minimal polynomial is $s$ or less.

Therefore, the degree of $g(X)$ is at most $2st$, and hence the number of parity-check digits of the code generated by $g(X)$ is no more than $2st$.

For $q=2$, we obtain the binary BCH codes.
t-error-correcting Reed-Solomon code with symbols from GF(q):
- \( s = 1 \), Block length: \( n = q - 1 \), \# parity-check digits: \( n - k = 2t \),
- Minimum distance: \( d_{\text{min}} = 2t + 1 \).

Reed-Solomon codes with code symbols from GF(2^m) (i.e., \( q = 2^m \))
- Let \( \alpha \) be a primitive element in GF(2^m)
- The generator polynomial of code length \( 2^m - 1 \) is
  \[
  g(x) = (x + \alpha)(x + \alpha^2) \ldots (x + \alpha^{2t}) = g_0 + g_1 x + \ldots + g_{2t-1} x^{2t-1} + x^{2t}
  \]
- \((n, n-2t)\) cyclic code and coefficients of \( g(X) \) is from GF(2^m)
- Let Message: \( a(X) = a_0 + a_1 X + a_2 X^2 + \cdots + a_{k-1} X^{k-1} \)
- In systematic form, the \( 2t \) parity-check digits are coefficients of
  \[
  b(X) = R_{g(X)} \left[ X^{2t} a(X) \right] = b_0 + b_1 X + \cdots + b_{2t-1} X^{2t-1}
  \]
Nonbinary BCH Codes and Reed-Solomon Codes

Hardware implementation
Nonbinary BCH Codes and Reed-Solomon Codes

Where

- \( \times \) multiplier in \( GF(2^m) \)
- \( + \) adder in \( GF(2^m) \)
- \( h \) storage device

Decoding of RS code:

\[
\begin{align*}
\mathbf{v}(x) &= v_0 + v_1 x + \ldots + v_{n-1} x^{n-1} \\
\mathbf{r}(x) &= r_0 + r_1 x + \ldots + r_{n-1} x^{n-1} \\
e(x) &= \mathbf{r}(x) - \mathbf{v}(x) = e_0 + e_1 x + \ldots + e_{n-1} x^{n-1} \\
e_i &= r_i - v_i, \quad \in GF(2^m)
\end{align*}
\]
Suppose the error pattern $e(x)$ contains $v$ errors (nonzero components), then

$$e(x) = e_{j_1}x^{j_1} + e_{j_2}x^{j_2} + \ldots + e_{j_v}x^{j_v}$$

To determine $e(x)$, we need error-locations $x^{j_i}$'s and the error values $e_{j_i}$'s (i.e. $v$ pairs($x^{j_i}$, $e_{j_i}$)'s).

In decoding a RS code, the same three steps used for decoding a binary BCH code are required; in addition, a fourth step involving calculation of error values is required.
Let $\beta_\ell = \alpha^{i\ell}$ \quad $\ell = 1.2,\ldots,n$

$s_1 = r(\alpha) = e_1\beta_1 + e_2\beta_2 + \ldots + e_v\beta_v$

$s_2 = r(\alpha^2) = e_1\beta_1^2 + \ldots + e_v\beta_v^2$

$\vdots \quad \vdots$

$s_{2t} = r(\alpha^{2t}) = e_1\beta_1^{2t} + \ldots + e_v\beta_v^{2t}$

note: $S_i = R_{(x + \alpha^i)}[r(x)] = b_i$

(i.e. $r(x) = e_i(x)(x + \alpha^i) + b_i$)
Nonbinary BCH Codes and Reed-Solomon Codes

- Syndrome computation circuit for R-S codes

(a) over $GF(2^m)$

(b) in binary form

Division circuit
Nonbinary BCH Codes and Reed-Solomon Codes

To find $\sigma(x)$: the error-location poly. by Berlekamp’s iterative algorithm

$$\sigma(x) = (1 + \beta_1 x)(1 + \beta_2 x)\ldots(1 + \beta_v x)$$

$$= 1 + \sigma_1 x + \ldots + \sigma_v x^v$$

Once $\sigma(x)$ is found, we can determine the error values.

Let $z(x) = 1 + (s_1 + \sigma_1)x + (s_2 + \sigma_1 s_1 + \sigma_2)x^2 + \ldots$

$$+ (s_v + \sigma_1 s_{v-1} + \ldots + \sigma_v)x^2$$

It can be shown that the error value at location $\beta_i = \alpha^{jl}$ is given by:

$$e_{j_i} = \frac{Z(\beta_l^{-1})}{\prod_{i=1}^v (1 + \beta_i \beta_l^{-1})}$$
Ex: Consider a triple-error-correcting Reed-Solomon code with symbols from GF(2^4)

The generator polynomial of this code is:

\[ g(x) = (x + \alpha)(x + \alpha^2) \ldots (x + \alpha^6) \]
\[ = \alpha^6 + \alpha^9 x + \alpha^6 x^2 + \alpha^4 x^3 + \alpha^{14} x^4 + \alpha^{10} x^5 + x^6 \]

(15, 9, 3) RS code, \( n = 15 \). \( n - k = 6, \ t = 3 \)

\[ x \quad (00\ldots 0) \quad \oplus \quad r = (000\alpha^7 00\alpha^3 000000\alpha^4 00) \]

\[ \therefore \ r(x) = \alpha^7 x^3 + \alpha^3 x^6 + \alpha^4 x^{12} \]
Nonbinary BCH Codes and Reed-Solomon Codes

Step 1. Using table 2.8 below to compute syndrome components:

<table>
<thead>
<tr>
<th>Power representation</th>
<th>Polynomial representation</th>
<th>4-Tuple representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(0 0 0 0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(1 0 0 0)</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>(0 1 0 0)</td>
</tr>
<tr>
<td>$\alpha^2$</td>
<td>$\alpha^2$</td>
<td>(0 0 1 0)</td>
</tr>
<tr>
<td>$\alpha^3$</td>
<td>$\alpha^3$</td>
<td>(0 0 0 1)</td>
</tr>
<tr>
<td>$\alpha^4$</td>
<td>$1 + \alpha$</td>
<td>(1 1 0 0)</td>
</tr>
<tr>
<td>$\alpha^5$</td>
<td>$\alpha + \alpha^2$</td>
<td>(0 1 1 0)</td>
</tr>
<tr>
<td>$\alpha^6$</td>
<td>$\alpha^2 + \alpha^3$</td>
<td>(0 0 1 1)</td>
</tr>
<tr>
<td>$\alpha^7$</td>
<td>$1 + \alpha + \alpha^3$</td>
<td>(1 1 0 1)</td>
</tr>
<tr>
<td>$\alpha^8$</td>
<td>$1 + \alpha^2 + \alpha^3$</td>
<td>(1 0 1 0)</td>
</tr>
<tr>
<td>$\alpha^9$</td>
<td>$\alpha + \alpha^3$</td>
<td>(0 1 0 1)</td>
</tr>
<tr>
<td>$\alpha^{10}$</td>
<td>$1 + \alpha + \alpha^2$</td>
<td>(1 1 1 0)</td>
</tr>
<tr>
<td>$\alpha^{11}$</td>
<td>$\alpha + \alpha^2 + \alpha^3$</td>
<td>(0 1 1 1)</td>
</tr>
<tr>
<td>$\alpha^{12}$</td>
<td>$1 + \alpha + \alpha^2 + \alpha^3$</td>
<td>(1 1 1 1)</td>
</tr>
<tr>
<td>$\alpha^{13}$</td>
<td>$1 + \alpha^2 + \alpha^3$</td>
<td>(1 0 1 1)</td>
</tr>
<tr>
<td>$\alpha^{14}$</td>
<td>$1 + \alpha^3$</td>
<td>(1 0 0 1)</td>
</tr>
</tbody>
</table>
Nonbinary BCH Codes and Reed-Solomon Codes

\[ S_1 = r(\alpha) = \alpha^{10} + \alpha^9 + \alpha = \alpha^{12} \quad \text{(Note, } \neq S_1^2) \]
\[ S_2 = r(\alpha^2) = \alpha^{13} + 1 + \alpha^{13} = 1 \]
\[ S_3 = r(\alpha^3) = \alpha + \alpha^6 + \alpha^{10} = \alpha^{14} \]
\[ S_4 = r(\alpha^4) = \alpha^4 + \alpha^{12} + \alpha^7 = \alpha^{10} \]
\[ S_5 = r(\alpha^5) = \alpha^7 + \alpha^3 + \alpha^4 = 0 \]
\[ S_6 = r(\alpha^6) = \alpha^{10} + \alpha^9 + \alpha = \alpha^{12} \]

Step 2. Find \[ \sigma(x) = 1 + \alpha^7 x + \alpha^4 x^2 + \alpha^6 x^3 \] by filling out table in next slide.
## Nonbinary BCH Codes and Reed-Solomon Codes

<table>
<thead>
<tr>
<th>$\mu$</th>
<th>$\sigma(\mu)(X)$</th>
<th>$d_\mu$</th>
<th>$l_\mu$</th>
<th>$\mu - l_\mu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$\alpha^{12}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$1 + \alpha^{12}X$</td>
<td>$\alpha^7$</td>
<td>1</td>
<td>0 (take $\rho = -1$)</td>
</tr>
<tr>
<td>2</td>
<td>$1 + \alpha^{3}X$</td>
<td>1</td>
<td>1</td>
<td>1 (take $\rho = 0$)</td>
</tr>
<tr>
<td>3</td>
<td>$1 + \alpha^{3}X + \alpha^{3}X^2$</td>
<td>$\alpha^7$</td>
<td>2</td>
<td>1 (take $\rho = 0$)</td>
</tr>
<tr>
<td>4</td>
<td>$1 + \alpha^{4}X + \alpha^{12}X^2$</td>
<td>$\alpha^{10}$</td>
<td>2</td>
<td>2 (take $\rho = 2$)</td>
</tr>
<tr>
<td>5</td>
<td>$1 + \alpha^{7}X + \alpha^{4}X^2 + \alpha^{6}X^3$</td>
<td>0</td>
<td>3</td>
<td>2 (take $\rho = 3$)</td>
</tr>
<tr>
<td>6</td>
<td>$1 + \alpha^{7}X + \alpha^{4}X^2 + \alpha^{6}X^3$</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
Step 3. By substituting $1, \alpha, \alpha^2, \ldots \alpha^{14}$ into $\sigma(x)$

$\Rightarrow \alpha^3, \alpha^9, \alpha^{12}$ are roots of $\sigma(x)$

$\therefore \alpha^{-3} = \alpha^{12}, \alpha^{-9} = \alpha^6, \alpha^{-12} = \alpha^3$ are the error-location numbers of $e(x)$  \(\therefore e(x) = e_3x^3 + e_6x^6 + e_{12}x^{12}\)

Step 4. From 6.34, find $z(x) = 1 + \alpha^2x + x^2 + \alpha^6x^3$

$\therefore e_3 = \frac{z(\alpha^{-3})}{(1 + \alpha^6\alpha^{-3})(1 + \alpha^{12}\alpha^{-3})} = \alpha^7$

$e_6 = \frac{z(\alpha^{-6})}{(1 + \alpha^3\alpha^{-6})(1 + \alpha^{12}\alpha^{-6})} = \alpha^3$
Nonbinary BCH Codes and Reed-Solomon Codes

\[ e_{12} = \frac{z(\alpha^{-12})}{(1 + \alpha^3 \alpha^{-12})(1 + \alpha^6 \alpha^{-12})} = \alpha^4 \]

\[ \therefore e(x) = \alpha^4 x^3 + \alpha^3 x^6 + \alpha^4 x^{12} \]

\[ \therefore v(x) = r(x) - e(x) = 0 \]

- If \( \beta \) is not a primitive element of \( \text{GF}(2^m) \), then the \( 2^m \)-ary code generated by
  \[ g(x) = (x + \beta)(x + \beta^2)\ldots(x + \beta^{2^t}) \]
  is a nonprimitive \( t \)-error-correcting RS code. The length \( n \) of this code is simply the order of \( \beta \).

- Decoding of a nonprimitive Reed-Solomon code is identical to the decoding of a primitive Reed-Solomon code.
Reed-Solomon codes are very effective for correcting multiple bursts of errors.

Two information symbols can be added to a RS code of length $n$ without reducing its minimum distance.

The extended RS code has length $n+2$ and the same number of parity-check symbols as the original code.

For a $t$-error-correcting RS code, the parity-check matrix may take the form

$$
H = \begin{bmatrix}
1 & \alpha & \alpha^2 & \ldots & \alpha^{n-1} \\
1 & \alpha^2 & (\alpha^2)^2 & \ldots & (\alpha^2)^{n-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \alpha^{2t} & (\alpha^{2t})^2 & \ldots & (\alpha^{2t})^{n-1}
\end{bmatrix}
$$
The parity-check matrix of the extended RS code is

\[
H_1 = \begin{bmatrix}
0 & 1 \\
0 & 0 \\
. & . \\
. & . \\
0 & 0 \\
0 & 0 \\
1 & 0 \\
\end{bmatrix}
\]
Weight Distribution and Error Detection of Binary BCH Codes
Weight Distribution and Error Detection of Binary BCH Codes

- The weight distributions of double-error-correcting, triple-error-correcting, and some low-rate primitive BCH codes have been completely determined.
- However, for the other BCH codes, their weight distribution are still unknown.
- Computation of the weight distribution of a double-error-correcting or a triple-error-correcting primitive BCH code can be achieved by first computing the weight distributions of its dual code and then applying the Macwilliams identity of (3.32).
The weight distribution of the dual of a double-error-correcting primitive BCH code of length $2^m - 1$ is given below:

**TABLE 6.9** WEIGHT DISTRIBUTION OF THE DUAL OF A DOUBLE-ERROR-CORRECTING PRIMITIVE BINARY BCH CODE OF LENGTH $2^m - 1$

<table>
<thead>
<tr>
<th>Weight, $i$</th>
<th>Number of vectors with weight $i$, $B_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$1$</td>
</tr>
<tr>
<td>$2^m - 1 - 2^{(m+1)/2 - 1}$</td>
<td>$[2^{m-2} + 2^{(m-1)/2 - 1}](2^m - 1)$</td>
</tr>
<tr>
<td>$2^m - 1$</td>
<td>$(2^m - 2^{m-1} + 1)(2^m - 1)$</td>
</tr>
<tr>
<td>$2^m - 1 + 2^{(m+1)/2 - 1}$</td>
<td>$[2^{m-2} - 2^{(m-1)/2 - 1}](2^m - 1)$</td>
</tr>
</tbody>
</table>
### TABLE 6.10  WEIGHT DISTRIBUTION OF THE DUAL OF A DOUBLE-ERROR-CORRECTING PRIMITIVE BINARY BCH CODE OF LENGTH $2^m - 1$

<table>
<thead>
<tr>
<th>Weight, $i$</th>
<th>Even $m \geq 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$1$</td>
</tr>
<tr>
<td>$2m - 1$</td>
<td>$2^{(m-2)/2-1}[2^{(m-2)/2} + 1](2^m - 1)/3$</td>
</tr>
<tr>
<td>$2m - 1$</td>
<td>$2^{(m+2)/2-1}(2^{m/2} + 1)(2^m - 1)/3$</td>
</tr>
<tr>
<td>$2m - 1$</td>
<td>$(2^{m-2} + 1)(2^m - 1)$</td>
</tr>
<tr>
<td>$2m - 1 + 2m/2 - 1$</td>
<td>$2^{(m+2)/2-1}(2^{m/2} - 1)(2^m - 1)/3$</td>
</tr>
<tr>
<td>$2m - 1 + 2^{(m+2)/2} - 1$</td>
<td>$2^{(m-2)/2-1}[2^{(m-2)/2} - 1](2^m - 1)/3$</td>
</tr>
</tbody>
</table>
### Table 6.11: Weight Distribution of the Dual of a Triple-Error-Correcting Primitive Binary BCH Code of Length $2^m - 1$

<table>
<thead>
<tr>
<th>Weight, $i$</th>
<th>Number of vectors with weight $i$, $B_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$2^{(m-5)/2}[2^{(m-3)/2} + 1](2^{m-1} - 1)(2^m - 1)/3$</td>
</tr>
<tr>
<td>$2^{m-1}$</td>
<td>$2^{(m-3)/2}[2^{(m-1)/2} + 1](5 \cdot 2^{m-1} + 4)(2^m - 1)/3$</td>
</tr>
<tr>
<td>$2^{m-1}$</td>
<td>$(9 \cdot 2^{2m-4} + 3 \cdot 2^{m-3} + 1)(2^m - 1)$</td>
</tr>
<tr>
<td>$2^{m-1} + 2^{(m-1)/2}$</td>
<td>$2^{(m-3)/2}[2^{(m-1)/2} - 1](5 \cdot 2^{m-1} + 4)(2^m - 1)/3$</td>
</tr>
<tr>
<td>$2^{m-1} + 2^{(m+1)/2}$</td>
<td>$2^{(m-5)/2}[2^{(m-3)/2} - 1](2^{m-1} - 1)(2^m - 1)/3$</td>
</tr>
</tbody>
</table>

### Table 6.12: Weight Distribution of the Dual of a Triple-Error-Correcting Primitive Binary BCH Code of Length $2^m - 1$

<table>
<thead>
<tr>
<th>Weight, $i$</th>
<th>Number of vectors with weight $i$, $B_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$[2^{m-1} + 2^{(m+4)/2 - 1}](2^m - 4)(2^m - 1)/960$</td>
</tr>
<tr>
<td>$2^{m-1}$</td>
<td>$7[2^{m-1} + 2^{(m+2)/2 - 1}](2^m - 1)/48$</td>
</tr>
<tr>
<td>$2^{m-1}$</td>
<td>$2(2^{m-1} + 2^{m/2 - 1})(3 \cdot 2^m + 8)(2^m - 1)/15$</td>
</tr>
<tr>
<td>$2^{m-1}$</td>
<td>$(29 \cdot 2^{2m} - 4 \cdot 2^m + 64)(2^m - 1)/64$</td>
</tr>
<tr>
<td>$2^{m-1} + 2^{m/2 - 1}$</td>
<td>$2(2^{m-1} - 2^{m/2 - 1})(3 \cdot 2^m + 8)(2^m - 1)/15$</td>
</tr>
<tr>
<td>$2^{m-1} + 2^{(m+2)/2 - 1}$</td>
<td>$7[2^{m-1} - 2^{(m+2)/2 - 1}](2^m - 1)/48$</td>
</tr>
<tr>
<td>$2^{m-1} + 2^{(m+4)/2 - 1}$</td>
<td>$[2^{m-1} - 2^{(m+4)/2 - 1}](2^m - 4)(2^m - 1)/960$</td>
</tr>
</tbody>
</table>
If a double-error-correcting or a triple-error-correcting primitive BCH code is used for error detection on a BSC with transition probability $p$, its probability of an undetected error can be computed from (3.36) and one of the weight distribution tables.

It would be interesting to know how a general $t$-error-correcting primitive BCH code performs when it’s used for error detection on a BSC with transition probability $p$. For a $t$-error-correcting primitive BCH code of length $2^m - 1$, if the number of parity-check digits is equal to $mt$ and $m$ is greater than certain constant $m_0(t)$, the number of code vectors of weight $i$ satisfies the following equalities:

$$A_t = \begin{cases} 
0 & \text{for } 0 < i \leq 2t \\
(1 + \lambda_0 \cdot n^{-1/10}) \binom{n}{i} 2^{-(n-k)} & \text{for } i > 2t 
\end{cases}$$
Weight Distribution and Error Detection of Binary BCH Codes

where \( n = 2^m - 1 \) and \( \lambda_0 \) is upper bounded by a constant.

From (3.19) and (6.26), we obtain

\[
P_u(E) = (1 + \lambda_0 \cdot n^{-1/10}) 2^{-(n-k)} \sum_{i=2t+1}^{n} \binom{n}{i} p^i (1 - p)^{n-i}
\]

Let \( \varepsilon = (2t + 1) / n \). Then

\[
\sum_{i=\varepsilon}^{n} \binom{n}{i} p^i (1 - p)^{n-i} \leq 2^{-nE(\varepsilon, p)}
\]

provided that \( p < \varepsilon \), where

\[
E(\varepsilon, p) = H(p) + (\varepsilon - p)H'(p) - H(\varepsilon) \Rightarrow > 0 \text{ for } \varepsilon > p
\]

\[
H(x) = -x \log_2 x - (1-x) \log_2 (1-x)
\]

\[
H'(x) = \log_2 \frac{1-x}{x}
\]
So we can obtain
\[ P_u(E) = (1 + \lambda_0 n^{-1/10}) 2^{-nE(\varepsilon, p)} 2^{-(n-k)} \]
For \( p < \varepsilon \) and sufficient large \( n \), \( P_u(E) \) can be made very small.
For \( p \geq \varepsilon \), we obtain upper bound. It’s clear from
\[
P_u(E) = (1 + \lambda_0 \cdot n^{-1/10}) 2^{-(n-k)} \sum_{i=0}^{n} \binom{n}{i} p^i (1 - p)^{n-i}
\]
Since
\[
\sum_{i=0}^{n} \binom{n}{i} p^i (1 - p)^{n-i} = 1
\]
so
\[
P_u(E) \leq (1 + \lambda_0 \cdot n^{-1/10}) 2^{-(n-k)}
\]
Weight Distribution and Error Detection of Binary BCH Codes

For a $t$-error-correcting primitive BCH code of length $2^m - 1$ with number of parity-check digits $n-k = mt$ and $m \geq m_0(t)$, its probability $p$ satisfies:

$$P_u(E) \leq \begin{cases} 
(1 + \lambda_0 n^{-1/10}) 2^{-n[1-R+E(\varepsilon,p)]} & \text{for } p < \varepsilon \\
(1 + \lambda_0 n^{-1/10}) 2^{-n(1-R)} & \text{for } p \geq \varepsilon
\end{cases}$$

where $\varepsilon = (2t + 1) / n$, $R = k / n$, and $\lambda_0$ is a constant.

In the nonbinary case, for a $t$-error-correcting primitive BCH code of length $q-1$ with symbols from GF$(q)$, the number of code vectors of weight $j$ is

$$A_j = \left( q - 1 \right)^{j-2t-1} \sum_{i=0}^{j-2t-1} (-1)^i \binom{j}{i} (q^{j-2t-i} - 1).$$