# Definition of BCH codes

BCH codes are cyclic codes over $\mathrm{GF}(q)$ (the *channel alphabet*) that are defined by a $(d-1) \times n$ check matrix over $\mathrm{GF}(q^m)$ (the *decoder alphabet*):

$$H = \begin{bmatrix} 1 & \alpha^b & \alpha^{2b} & \cdots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \cdots & \alpha^{(n-1)(b+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{b+d-2} & \alpha^{2(b+d-2)} & \cdots & \alpha^{(n-1)(b+d-2)} \end{bmatrix}$$

Design parameters:

- $\alpha$ is an element of $\mathrm{GF}(q^m)$ of order $n$
- $b$ is any integer ($0 \le b < n$ is sufficient)
- $d$ is an integer with $2 \le d \le n$ ($d = 1$, $d = n + 1$ are trivial cases)

Rows of $H$ are the first $n$ powers of *consecutive* powers of $\alpha$.

# Special cases of BCH codes

A *primitive BCH code* is a BCH code defined using a primitive element $\alpha$.

If $\alpha$ is a primitive element of $\mathrm{GF}(q^m)$, then the blocklength is $n = q^m - 1$.
This is the maximum possible blocklength for decoder alphabet $\mathrm{GF}(q^m)$.

A *narrow-sense BCH code* is a BCH code with $b = 1$.
Some decoding formulas simplify when $b = 1$. But $b \neq 1$ is usually used.

Parity-check matrix for a $t$-error-correcting primitive narrow-sense BCH code where $\alpha$ is an $n$-th root of unity in $\mathrm{GF}(q^m)$ and $n = q^m - 1$:

$$\begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{(n-1)} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{2t} & \alpha^{4t} & \cdots & \alpha^{2t(n-1)} \end{bmatrix}$$

Each row of $H$ is a row of the finite field Fourier transform matrix.

Codewords are $n$-tuples whose spectra have $0$'s at $2t$ consecutive frequencies.

# Reed-Solomon codes

Reed-Solomon codes are special case of BCH codes where decoder alphabet is channel alphabet.

Minimal polynomials over $\mathrm{GF}(Q)$ of elements of $\mathrm{GF}(Q)$ have degree 1.

The generator polynomial of a $t$-error-correcting Reed-Solomon code is

$$\begin{aligned} g(x) &= (x-\alpha^b)(x-\alpha^{b+1})\cdots(x-\alpha^{b+2t-1}) \\ &= g_0 + g_1 x + \cdots + g_{2t-1}x^{2t-1} + x^{2t}\,, \end{aligned}$$

where $g_0, g_1 \ldots, g_{2t-1}$ are elements of $\mathrm{GF}(Q)$.

The minimum distance is $2t+1$, independent of the choice of $\alpha$ and $b$.

Usually $\alpha$ is chosen to be primitive in order to maximize blocklength.

The base exponent $b$ can be chosen to reduce encoder/decoder complexity.

- $b = 0$ results in overall parity-check equation
- $b = 2^{m-1}$ corresponds to reversible generator polynomial

# Reed-Solomon code: example

Audio CDs and CD-ROMs use 2EC Reed-Solomon codes over $\mathrm{GF}(2^8)$.

Field arithmetic is defined by primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$.

The base exponent is $b = 0$.

The generator polynomial of the (255,251) Reed-Solomon code is

$$\begin{aligned} g(x) &= (x+1)(x+\alpha)(x+\alpha^2)(x+\alpha^3) \\ &= x^4 + \alpha^{75}x^3 + \alpha^{249}x^2 + \alpha^{78}x + \alpha^6 \\ &= x^4 + \mathtt{0f}x^3 + \mathtt{36}x^2 + \mathtt{78}x + \mathtt{40}\,. \end{aligned}$$

Most significant bit is on the *left* in hexadecimal representations of coefficients; that is, $1 = \mathtt{01}$, $\alpha = \mathtt{02}$, $\alpha^2 = \mathtt{04}$, and so on.

There are no prime trinomials of degree 8; in fact, there are no prime trinomials of degree $8m$ for any $m$.

# (15,7,5) BCH code: parity-check matrix

Parity-check matrix of $(15, 7, 5)$ binary BCH code:

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{14} \\ 1 & \alpha^3 & \alpha^9 & \cdots & \alpha^{42} \end{bmatrix} = \left[\begin{array}{cccccccccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{array}\right]$$

Rows of $H$ are linearly independent, so there are $2^8$ syndromes. There are

$$1 + \binom{15}{1} + \binom{15}{2} = 121 < 2^7 < 2^8$$

error patterns of weight $2$. This code does *not* achieve Hamming bound.

Systematic parity-check matrix can be found from the generator polynomial.

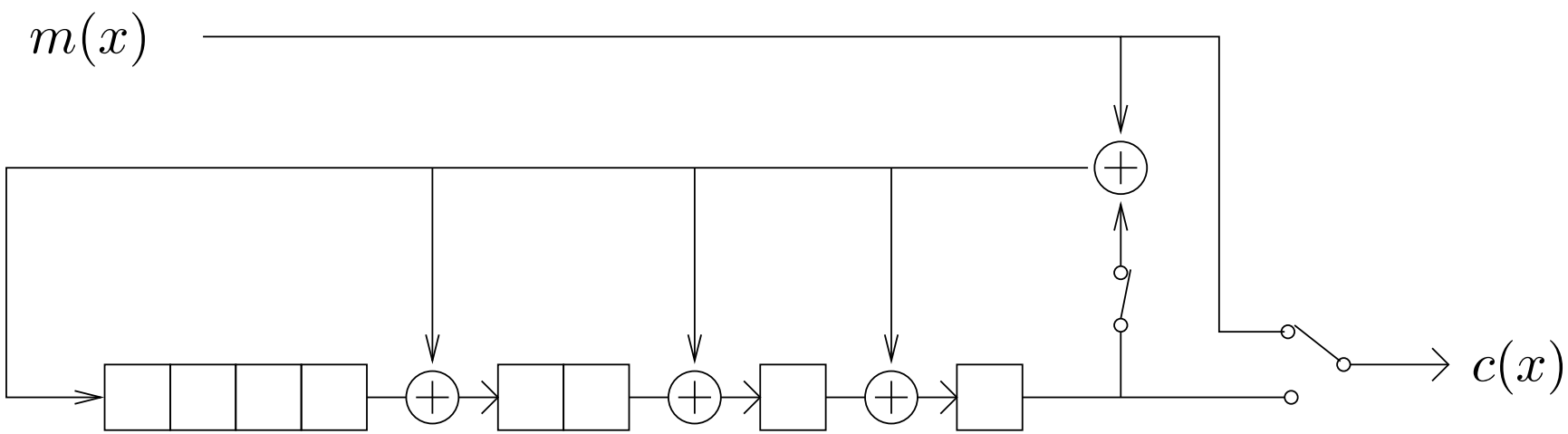
There is no (15,8) binary linear block code with minimum distance $5$.

# (15,7,5) BCH code: generator polynomial

Generator polynomial is LCM of minimal polynomials of $\alpha$, $\alpha^2$, $\alpha^3$, $\alpha^4$ :

$$g(x) = f_1(x)f_3(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)$$
$$= x^8 + x^7 + x^6 + x^4 + 1 = 1 + x^4 + x^6 + x^7 + x^8$$

BCH codes are cyclic and have shift register encoders and syndrome circuits:




Modified error trapping can be used for $(15, 7, 5)$ binary BCH code.

Any $2$-bit error pattern can be rotated into the $8$ check positions. However, two error trapping passes may be needed.

# (15,5,7) BCH code

Generator polynomial of 3EC BCH code is defined by zeroes $\alpha$, $\alpha^3$, $\alpha^5$:

$$g(x) = f_1(x)f_3(x)f_5(x) = (x^4+x+1)(x^4+x^3+x^2+x+1)(x^2+x+1)$$
$$= x^{10}+x^8+x^5+x^4+x^2+x+1$$

Parity-check matrix is $3 \times 15$ over $\mathrm{GF}(2^4)$ or $12 \times 15$ over $\mathrm{GF}(2)$:

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{14} \\ 1 & \alpha^3 & \alpha^6 & \cdots & \alpha^{42} \\ 1 & \alpha^5 & \alpha^{10} & \cdots & \alpha^{70} \end{bmatrix} =$$

$$\left[\begin{array}{cccccccccccccc} 1&0&0&0&1&0&0&1&1&0&1&0&1&1&1 \\ 0&1&0&0&1&1&0&1&0&1&1&1&1&0&0 \\ 0&0&1&0&0&1&1&0&1&0&1&1&1&1&0 \\ 0&0&0&1&0&0&1&1&0&1&0&1&1&1&1 \\ \hline 1&0&0&0&1&1&0&0&0&1&1&0&0&0&1 \\ 0&0&0&1&1&0&0&0&1&1&0&0&0&1&1 \\ 0&0&1&0&1&0&0&1&0&1&0&0&1&0&1 \\ 0&1&1&1&1&0&1&1&1&1&0&1&1&1&1 \\ \hline 1&0&1&1&0&1&1&0&1&1&0&1&1&0&1 \\ 0&1&1&0&1&1&0&1&1&0&1&1&0&1&1 \\ 0&1&1&0&1&1&0&1&1&0&1&1&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{array}\right]$$

The last two rows are linearly redundant $\Rightarrow$ 10 check equations $\Rightarrow$ $k = 5$.

# (15,5,7) BCH code: redundant rows

$H$ has two redundant rows:

- bottom row is zero
- next to last row is same as previous row

$H$ has 10 independent rows and defines a $(15, 5)$ binary cyclic code.

Parity-check polynomial $h(x) = (x^4 + x^3 + 1)(x + 1)$ includes all the prime divisors of $x^{15} - 1$ that are *not* included in $g(x)$.

The dual of this BCH code is a $(15, 10)$ expurgated Hamming code with $d^* = 4$.

The $(15, 5)$ BCH code is obtained from the $(15, 4)$ maximum-length code by augmentation — including the complements of the original codewords.

The weight enumerator of dual is $A(x) = 1 + 15x^7 + 15x^8 + x^{15}$.

Using the Macwilliams identity, we can find the weight enumerator of the

# (31,16,7) BCH code

Zeroes of codewords are $\alpha$, $\alpha^3$, $\alpha^5$ in $\mathrm{GF}(2^5)$. Parity-check matrix:

$$H = \left[\begin{array}{ccccccccccccccccccccccccccccc}
1&0&0&0&0&1&0&0&1&0&1&1&0&0&1&1&1&1&1&0&0&0&1&1&0&1&1&1&0&1&0\\
0&1&0&0&0&0&1&0&0&1&0&1&1&0&0&1&1&1&1&1&0&0&0&1&1&0&1&1&1&0&1\\
0&0&1&0&0&1&0&1&1&0&0&1&1&1&1&1&0&0&0&1&1&0&1&1&1&0&1&0&1&0&0\\
0&0&0&1&0&0&1&0&1&1&0&0&1&1&1&1&1&0&0&0&1&1&0&1&1&1&0&1&0&1&0\\
0&0&0&0&1&0&0&1&0&1&1&0&0&1&1&1&1&1&0&0&0&1&1&0&1&1&1&0&1&0&1\\
\hline
1&0&0&0&0&1&1&0&0&1&0&0&1&1&1&1&1&0&1&1&1&0&0&0&1&0&1&0&1&1&0\\
0&0&1&1&1&1&1&0&1&1&1&0&0&0&1&0&1&0&1&1&0&1&0&0&0&0&1&1&0&0&1\\
0&0&0&0&1&1&0&0&1&0&0&1&1&1&1&1&0&1&1&1&0&0&0&1&0&1&0&1&1&0&1\\
0&1&1&1&1&1&0&1&1&1&0&0&0&1&0&1&0&1&1&0&1&0&0&0&0&1&1&0&0&1&0\\
0&0&0&1&0&1&0&1&1&0&1&0&0&0&0&1&1&0&0&1&0&0&1&1&1&1&1&0&1&1&1\\
\hline
1&1&1&1&0&1&0&0&0&1&0&0&1&0&1&0&1&1&0&0&0&0&1&1&1&0&0&1&1&0&1\\
0&0&0&1&0&0&1&0&1&0&1&1&0&0&0&0&1&1&1&0&0&1&1&0&1&1&1&1&1&0&1\\
0&1&0&1&1&0&0&0&0&1&1&1&0&0&1&1&0&1&1&1&1&1&0&1&0&0&0&1&0&0&1\\
0&0&0&1&1&1&0&0&1&1&0&1&1&1&1&1&0&1&0&0&0&1&0&0&1&0&1&0&1&1&0\\
0&0&1&1&0&1&1&1&1&1&0&1&0&0&0&1&0&0&1&0&1&0&1&1&0&0&0&0&1&1&1
\end{array}\right]$$

# (31,16,7) BCH code (cont.)

Generator polynomial:

$$\begin{aligned} g(x) &= f_1(x)f_3(x)f_5(x) \\ &= (x^5+x^2+1)(x^5+x^4+x^3+x^2+1)(x^5+x^4+x^2+x+1) \\ &= x^{15}+x^{11}+x^{10}+x^9+x^8+x^7+x^5+x^3+x^2+x+1 \end{aligned}$$

It is *not* obvious that every set of $6$ columns of $H$ is linearly independent!

For blocklength 31, all binary BCH codes with $d^* = 7$ have 15 check bits.

The expanded code with $(n, k, d^*) = (32, 16, 8)$ is a Reed-Muller code.

# BCH codes with decoder alphabet GF(16)

Suppose that $\alpha$ is primitive in $\mathrm{GF}(16)$.

The following parity-check matrix defines a primitive, narrow-sense BCH code over each channel alphabet that is a subfield of $\mathrm{GF}(16)$.

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \cdots & \alpha^{14} \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \cdots & \alpha^{28} \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 & \cdots & \alpha^{42} \\ 1 & \alpha^4 & \alpha^8 & \alpha^{12} & \cdots & \alpha^{56} \end{bmatrix}$$

The three possible channel alphabets are $\mathrm{GF}(2)$, $\mathrm{GF}(2^2)$, and $\mathrm{GF}(2^4)$:

The BCH codes corresponding to these channels alphabets are

- (15,7) binary BCH code over $\mathrm{GF}(2)$ (presented earlier in lecture)
- (15,9) BCH code over $\mathrm{GF}(4)$
- (15,11) Reed-Solomon code over $\mathrm{GF}(16)$

Blocklengths *in symbols* are $15$; blocklengths *in bits* are $15$, $30$, and $60$.

# Channel alphabet GF(16)

Fact: the four rows of $H$ are linearly independent over $\mathrm{GF}(2^4)$.

Thus $H$ defines a $(15, 11)$ code over $\mathrm{GF}(2^4)$ with minimum distance $5$.

This code is a $(15, 11)$ 2EC *Reed-Solomon code* over $\mathrm{GF}(16)$.

Using table of powers of $\alpha$, we can calculate generator polynomial:

$$\begin{aligned} g(x) &= (x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4) \\ &= \alpha^{10} + \alpha^3 x + \alpha^6 x^2 + \alpha^{13} x^3 + x^4 \\ &= 7 + 8\,x + \mathtt{E}\,x^2 + \mathtt{D}\,x^3 + x^4 \end{aligned}$$

Coefficients of generator polynomial are computed using $\mathrm{GF}(2^4)$ arithmetic.

Coefficients can be expressed either in exponential or binary representation.

- exponential notation simplifies multiplication (add exponents mod $15$)
- binary notation simplifies addition (exclusive-or of $4$-bit values)

Hardware implementations use bit vectors and often log/antilog tables.

# Channel alphabet GF(4)

Let $\mathrm{GF}(4)$ be $\{0, 1, \beta, \delta\}$, where $\delta = \beta + 1 = \beta^2$.

$\mathrm{GF}(16)$ is defined by primitive polynomial $x^2 + x + \beta$ over $\mathrm{GF}(4)$.

$H$ defines a BCH code over subfield $\mathrm{GF}(4)$.

$$H_{[2^2]} = \left[\begin{array}{cccccccccccccc}
1 & 0 & \beta & \beta & 1 & \beta & 0 & \delta & \delta & \beta & \delta & 0 & 1 & 1 & \delta \\
0 & 1 & 1 & \delta & 1 & 0 & \beta & \beta & 1 & \beta & 0 & \delta & \delta & \beta & \delta \\
\hline
1 & \beta & 1 & 0 & \delta & \delta & 1 & \delta & 0 & \beta & \beta & \delta & \beta & 0 & 1 \\
0 & 1 & 1 & \beta & 1 & 0 & \delta & \delta & 1 & \delta & 0 & \beta & \beta & \delta & \beta \\
\hline
1 & \beta & 0 & \beta & 1 & 1 & \beta & 0 & \beta & 1 & 1 & \beta & 0 & \beta & 1 \\
0 & \delta & \beta & \beta & \delta & 0 & \delta & \beta & \beta & \delta & 0 & \delta & \beta & \beta & \delta \\
\hline
1 & 1 & \delta & 1 & 0 & \beta & \beta & 1 & \beta & 0 & \delta & \delta & \beta & \delta & 0 \\
0 & 1 & 1 & \delta & 1 & 0 & \beta & \beta & 1 & \beta & 0 & \delta & \delta & \beta & \delta
\end{array}\right]$$

The final two rows, corresponding to conjugate $\alpha^4$ over $\mathrm{GF}(4)$, are redundant. (Row 7 equals row 1 + row 2, while row 8 equals row 2).

Thus $g(x) = f_1(x)f_2(x)f_3(x)$ generates a $(15, 9, 5)$ code over $\mathrm{GF}(4)$.

# Channel alphabet GF(2)

This $16 \times 15$ matrix has redundant rows over $\mathrm{GF}(2)$. Every row in the second and fourth blocks is a linear combination of the first four rows.

$$H_{[2^1]} = \left[\begin{array}{ccccccccccccccc}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
\hline
1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1
\end{array}\right]$$

When we delete the redundant rows of $H$, we obtain the parity-check matrix of the $(15, 7)$ 2EC BCH code over $\mathrm{GF}(2)$ shown earlier.

# Vandermonde matrix

*Definition*: The $\mu \times \mu$ Vandermonde matrix $V(X_1, \ldots, X_\mu)$ is

$$V = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ X_1 & X_2 & \cdots & X_\mu \\ \vdots & \vdots & \ddots & \vdots \\ X_1^{\mu-1} & X_2^{\mu-1} & \cdots & X_\mu^{\mu-1} \end{bmatrix}.$$

One application of Vandermonde matrices is for polynomial interpolation. Given values of $f(x)$ of degree $\mu - 1$ at $\mu$ distinct points $X_1, \ldots, X_\mu$,

$$Y_i = f(X_i) = f_0 + f_1 X_i + \cdots + f_{\mu-1} X^{\mu-1} \quad (i = 1, \ldots, \mu)$$

the coefficients of $f(x)$ can be found by solving the matrix equation

$$\begin{bmatrix} Y_1 & Y_2 & \ldots & Y_\mu \end{bmatrix} = \begin{bmatrix} f_0 & f_1 & \ldots & f_{\mu-1} \end{bmatrix} V(X_1, \ldots, X_\mu).$$

The Vandermonde matrix is also defined using the transpose of the above matrix.

# Nonsingular Vandermonde matrix

*Lemma*: The Vandermonde matrix $V(X_1, \ldots, X_\mu)$ is nonsingular if and only if the parameters $X_1, \ldots, X_\mu$ are distinct. In fact,

$$\det V(X_1, \ldots, X_\mu) = \prod_{i>j} (X_i - X_j) = \prod_{i=1}^{\mu} \prod_{j=1}^{i-1} (X_i - X_j) .$$

*Proof*: The determinant is a polynomial in $\mu$ variables, $X_1, \ldots, X_\mu$.
As a polynomial in $X_i$, its zeroes are $X_j$ for $j \neq i$.
Thus $X_i - X_j$ is a factor of the determinant for every pair $(i, j)$ with $i > j$.

These are *all* the factors because the degree of $\det V(X_1, \ldots, X_\mu)$ is

$$0 + 1 + \cdots + (\mu - 2) + (\mu - 1) = \frac{\mu(\mu - 1)}{2} = \binom{\mu}{2} .$$

The coefficient of the main diagonal monomial

$$\prod_{i=1}^{\mu} X_i^{i-1} = 1 \cdot X_2 \cdot X_3^2 \cdot \ldots \cdot X_\mu^{\mu-1}$$

is $1$ in both the determinant and the above formula for the determinant.

# BCH bound

*Theorem*: A BCH code whose check matrix has $d-1$ rows has $d_{\mathsf{min}} \geq d$.

*Proof*: To show that every set of $d-1$ columns of $H$ is linearly independent over $\mathrm{GF}(q^m)$, consider the submatrix consisting of columns $i_1, \ldots, i_{d-1}$.

$$\det \begin{bmatrix} \alpha^{i_1 b} & \alpha^{(i_1+1)b} & \cdots & \alpha^{i_{d-1} b} \\ \alpha^{i_1 (b+1)} & \alpha^{(i_1+1)(b+1)} & \cdots & \alpha^{i_{d-1}(b+1)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{i_1(b+d-2)} & \alpha^{(i_1+1)(b+d-2)} & \cdots & \alpha^{i_{d-1}(b+d-2)} \end{bmatrix} =$$

$$(\alpha^{i_1 b}\alpha^{i_2 b} \cdots \alpha^{i_{d-1} b}) \det \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha^{i_1} & \alpha^{i_1+1} & \cdots & \alpha^{i_{d-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{i_1(d-2)} & \alpha^{(i_1+1)(d-2)} & \cdots & \alpha^{i_{d-1}(d-2)} \end{bmatrix} \neq 0$$

This determinant is nonzero because $\alpha^{i_1} \neq 0, \ldots, \alpha^{i_{d-1}} \neq 0$ and the second matrix is a *Vandermonde* matrix with distinct columns.

# Design of BCH codes

Codewords have zeroes that are $d-1$ consecutive powers of $\alpha$.
Conjugates over channel alphabet $\mathrm{GF}(q)$ are also zeroes.
The degree of the generator polynomial is the total number of conjugates.

*Example*: Channel alphabet $\mathrm{GF}(2)$, decoder alphabet $\mathrm{GF}(2^6)$.

The first six conjugacy classes, represented by exponents:

$$\{0\} \quad \{1,2,4,8,16,32\} \quad \{3,6,12,24,48,33\}$$
$$\{5,10,20,40,17,34\} \quad \{7,14,28,56,49,35\} \quad \{9,18,36\}$$

BCH code parameters:

- $d^* = 5$ requires 4 powers. Exponents $\{1,2,3,4\} \Rightarrow 12$ conjugates.
- $d^* = 9$ requires 8 powers. Exponents $\{1,\ldots,8\} \Rightarrow 24$ conjugates.
- $d^* = 11$ requires 10 powers. Exponents $\{1,\ldots,10\} \Rightarrow 27$ conjugates.
- $d^* = 4$ requires 3 powers.
  - Exponents $\{1,2,3\} \Rightarrow 12$ conjugates.
  - Better: $\{0,1,2\} \Rightarrow 7$ conjugates (expurgated code)

# GF(256): powers of primitive element $\alpha = 02$

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 01 | 02 | 04 | 08 | 10 | 20 | 40 | 80 | 1D | 3A | 74 | E8 | CD | 87 | 13 | 26 |
| 16 | 4C | 98 | 2D | 5A | B4 | 75 | EA | C9 | 8F | 03 | 06 | 0C | 18 | 30 | 60 | C0 |
| 32 | 9D | 27 | 4E | 9C | 25 | 4A | 94 | 35 | 6A | D4 | B5 | 77 | EE | C1 | 9F | 23 |
| 48 | 46 | 8C | 05 | 0A | 14 | 28 | 50 | A0 | 5D | BA | 69 | D2 | B9 | 6F | DE | A1 |
| 64 | 5F | BE | 61 | C2 | 99 | 2F | 5E | BC | 65 | CA | 89 | 0F | 1E | 3C | 78 | F0 |
| 80 | FD | E7 | D3 | BB | 6B | D6 | B1 | 7F | FE | E1 | DF | A3 | 5B | B6 | 71 | E2 |
| 96 | D9 | AF | 43 | 86 | 11 | 22 | 44 | 88 | 0D | 1A | 34 | 68 | D0 | BD | 67 | CE |
| 112 | 81 | 1F | 3E | 7C | F8 | ED | C7 | 93 | 3B | 76 | EC | C5 | 97 | 33 | 66 | CC |
| 128 | 85 | 17 | 2E | 5C | B8 | 6D | DA | A9 | 4F | 9E | 21 | 42 | 84 | 15 | 2A | 54 |
| 144 | A8 | 4D | 9A | 29 | 52 | A4 | 55 | AA | 49 | 92 | 39 | 72 | E4 | D5 | B7 | 73 |
| 160 | E6 | D1 | BF | 63 | C6 | 91 | 3F | 7E | FC | E5 | D7 | B3 | 7B | F6 | F1 | FF |
| 176 | E3 | DB | AB | 4B | 96 | 31 | 62 | C4 | 95 | 37 | 6E | DC | A5 | 57 | AE | 41 |
| 192 | 82 | 19 | 32 | 64 | C8 | 8D | 07 | 0E | 1C | 38 | 70 | E0 | DD | A7 | 53 | A6 |
| 208 | 51 | A2 | 59 | B2 | 79 | F2 | F9 | EF | C3 | 9B | 2B | 56 | AC | 45 | 8A | 09 |
| 224 | 12 | 24 | 48 | 90 | 3D | 7A | F4 | F5 | F7 | F3 | FB | EB | CB | 8B | 0B | 16 |
| 240 | 2C | 58 | B0 | 7D | FA | E9 | CF | 83 | 1B | 36 | 6C | D8 | AD | 47 | 8E | 01 |

# BCH codes with decoder alphabet GF(256)

Narrow-sense primitive 2EC BCH codes over $\mathrm{GF}(2)$, $\mathrm{GF}(2^2)$, $\mathrm{GF}(2^4)$, $\mathrm{GF}(2^8)$ can be defined by the same parity-check matrix:

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{254} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{508} \\ 1 & \alpha^3 & \alpha^6 & \cdots & \alpha^{752} \\ 1 & \alpha^4 & \alpha^8 & \cdots & \alpha^{1016} \end{bmatrix}$$

Generator polynomials $\mathrm{LCM}(f_1(x), \ldots, f_4(x))$ have subfield coefficients.

$$\mathrm{GF}(2^2) = \{0, 1, \alpha^{85}, \alpha^{170}\} = \{\texttt{00}, \texttt{01}, \texttt{D6}, \texttt{D7}\}$$

$$\mathrm{GF}(2^4) = \{0, 1, \alpha^{17}, \ldots, \alpha^{238}\} = \mathrm{span}\{\texttt{01}, \texttt{0B}, \texttt{98}, \texttt{D6}\}$$

| Subfield | Degree | Polynomial coefficients |
|---|---|---|
| $\mathrm{GF}(2^8)$ | 4 | `01 1E D8 E7 74` |
| $\mathrm{GF}(2^4)$ | 8 | `01 D6 01 DD 0B 98 98 98 D7` |
| $\mathrm{GF}(2^2)$ | 12 | `01 01 00 D7 00 00 00 D6 D7 D7 01 D7 01` |
| $\mathrm{GF}(2)$ | 16 | `1 0 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1` |

# Encoding and syndrome circuits

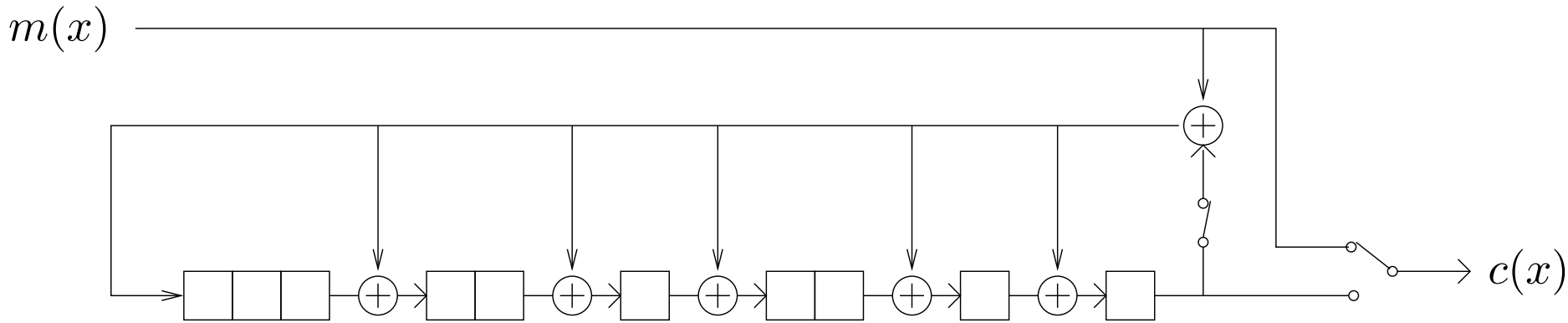
Binary BCH codes are defined using $\mathrm{GF}(2^m)$ but are cyclic over $\mathrm{GF}(2)$.

Shift registers can be used for encoding and for syndrome computation.

The $(31, 21)$ binary primitive 2EC BCH code with generator polynomial

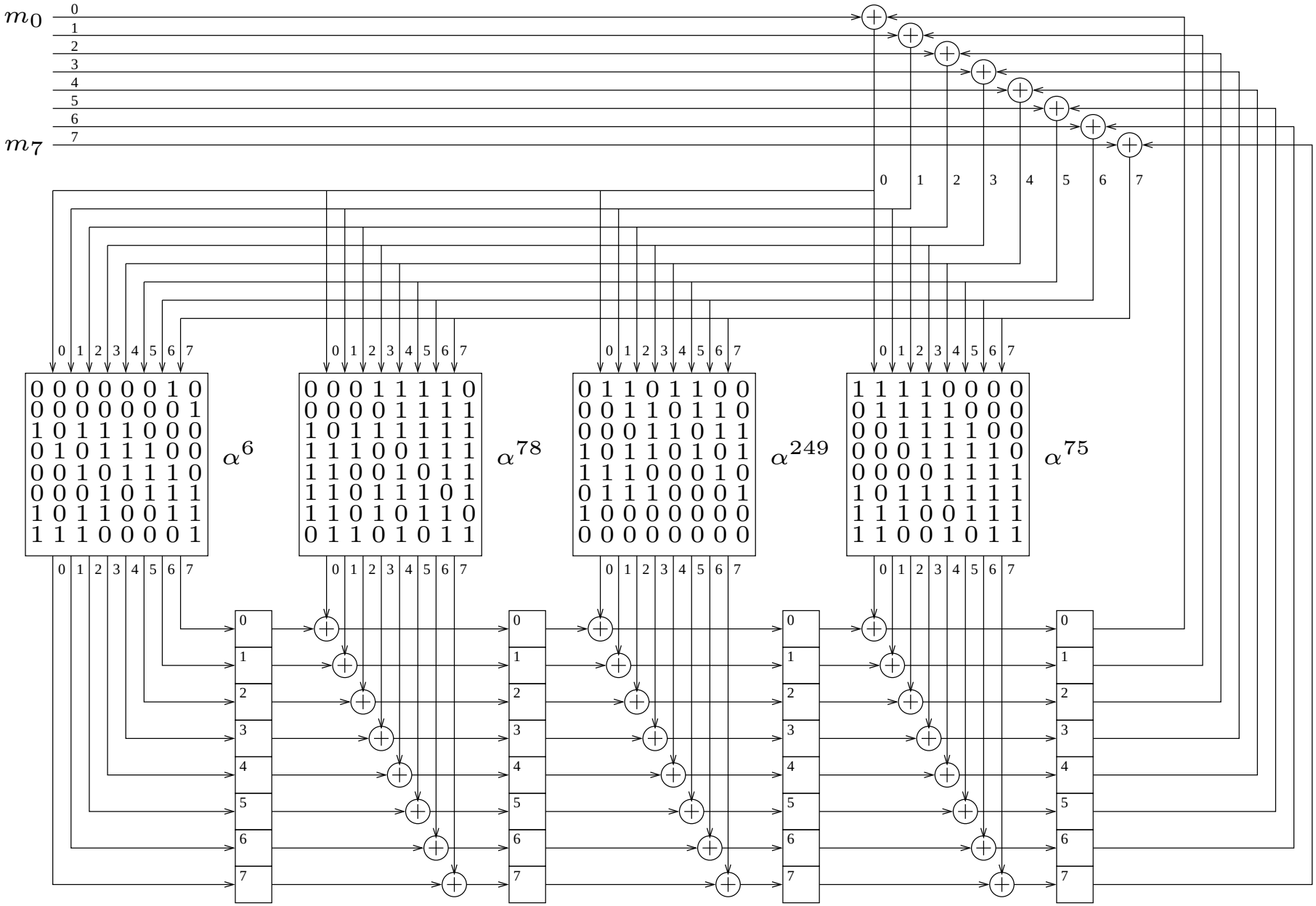
$$\begin{aligned} g(x) &= (x^5 + x^2 + 1)(x^5 + x^4 + x^3 + x^2 + 1) \\ &= 1 + x^3 + x^5 + x^6 + x^8 + x^9 + x^{10} \end{aligned}$$

has the following shift register encoder.




Syndrome $s(x) \bmod g(x)$ used for error detection has a similar circuit.

# Encoder for (255,251) Reed-Solomon code



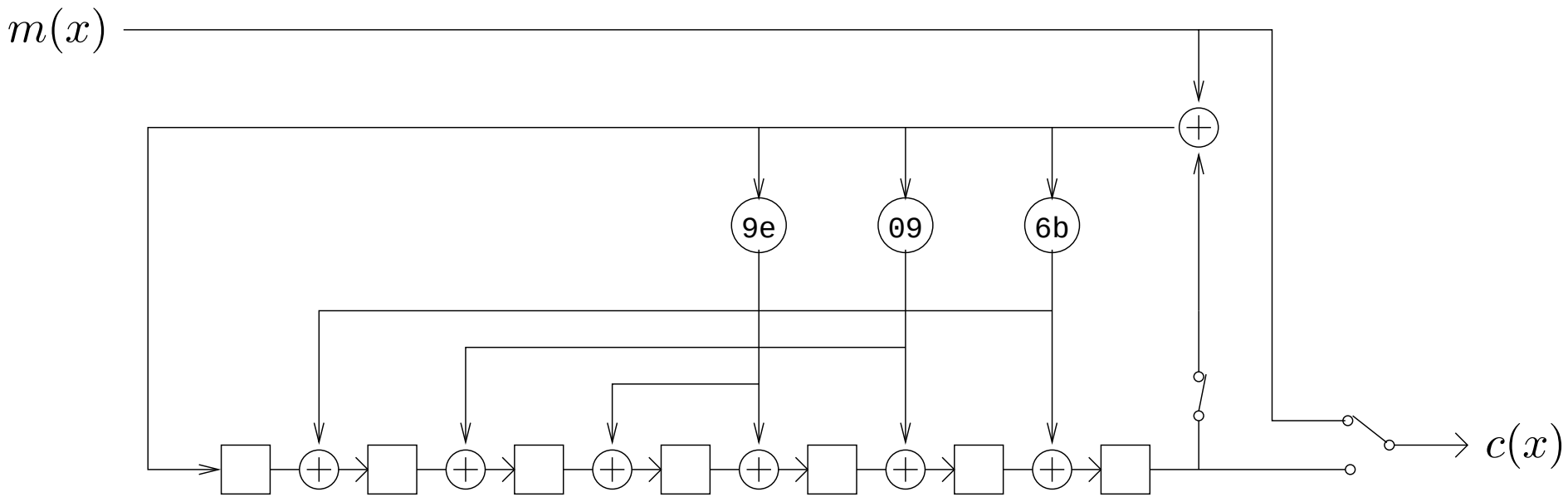
# $(255, 248)$ Reed-Solomon encoder

A Reed-Solomon code with $d^* = 8$ has the following generator polynomial:

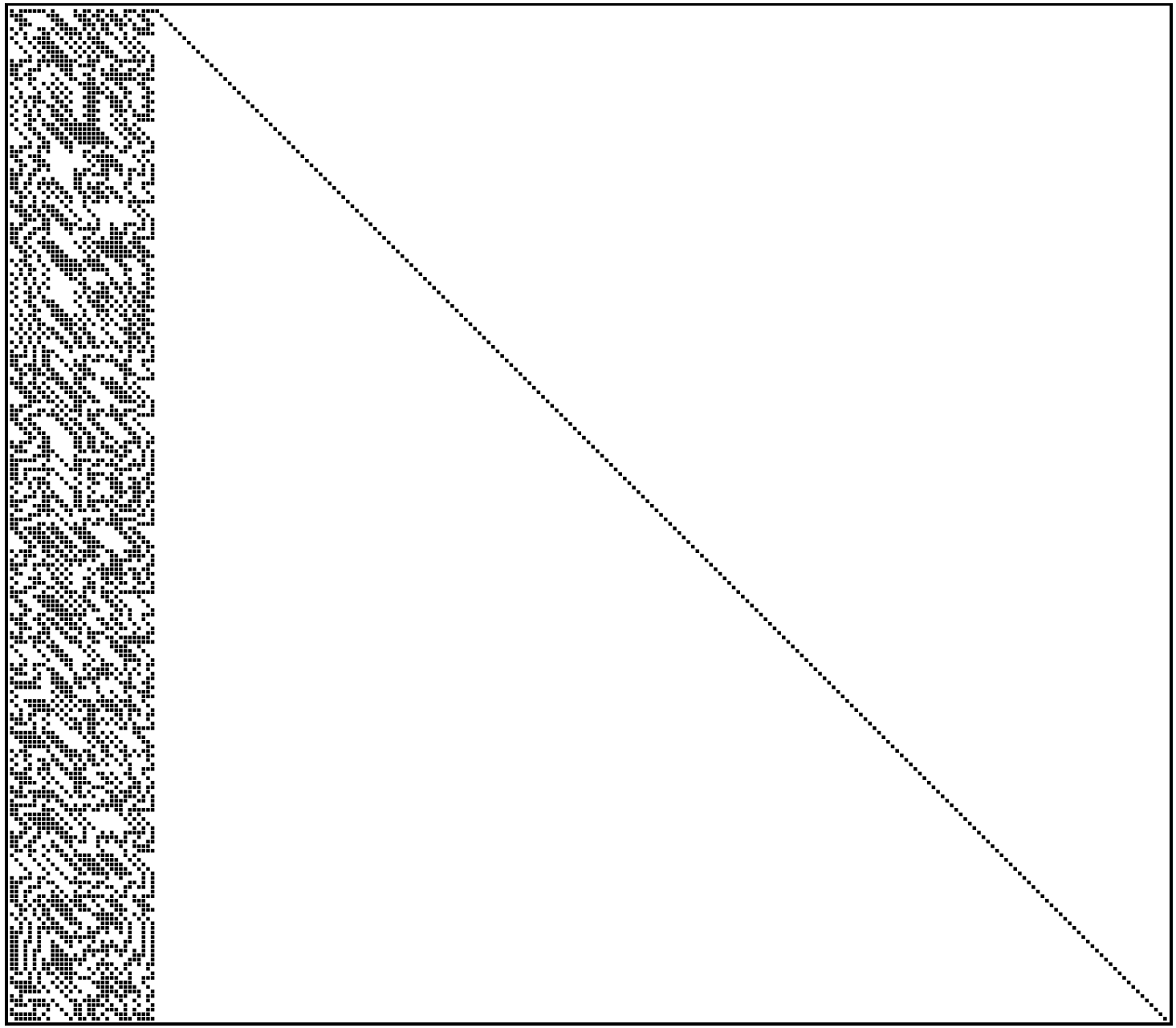
$$g(x) = (x + \alpha^{-3})(x + \alpha^{-2})(x + \alpha^{-1})(x + 1)(x + \alpha^{+1})(x + \alpha^{+2})(x + \alpha^{+3})$$
$$= x^7 + \mathtt{6b}x^6 + \mathtt{09}x^5 + \mathtt{9e}x^4 + \mathtt{9e}x^3 + \mathtt{09}x^2 + \mathtt{6b}x + 1$$

Since the reciprocals of its zeroes are also zeroes, $g(x)$ is its mirror image.

Thus the encoder corresponding to $g(x)$ has only $3$ distinct scalers.



# Generator matrix for $(255, 223)$ 4EC BCH code



# Decoding algorithms for BCH codes

Decoding BCH and Reed-Solomon codes consists of these four steps.

1. Compute partial syndromes $S_i = r(\alpha^i)$ for $i = b, \ldots, b+d-2$.
2. Find coefficients $\Lambda_1, \Lambda_2, \ldots, \Lambda_\nu$ of *error-locator polynomial*
$$\Lambda(x) = \prod_{i=1}^{\nu} (1 - x X_i)$$
by solving *linear* equations with constant coefficients $S_b, \ldots, S_{b+d-2}$.
3. Find the zeroes $X_1^{-1}, \ldots, X_\nu^{-1}$ of $\Lambda(x)$. If there are $\nu$ symbol errors, they are in locations $i_1, \ldots, i_\nu$ where $X_1 = \alpha^{i_1}, \ldots, X_\nu = \alpha^{i_\nu}$.
4. Solve linear equations with "constant" coefficients powers of $X_i$ for *error magnitudes* $Y_1, \ldots, Y_\nu$. (Only if $\mathrm{GF}(2)$ not channel alphabet.)

Efficient procedures for solving linear systems of equations in steps 2 and 4:

- Berlekamp, Massey (step 2)
- Forney (step 4)
- Sugiyama-Kasahara-Hirasawa-Namekawa ("Euclidean") (steps 2 and 4)

# Error locations and magnitudes

Suppose there are $\nu \leq t$ errors in locations $i_1, \ldots, i_\nu$.

Let error magnitudes be $e_{i_1}, \ldots, e_{i_\nu}$ (values in $\mathrm{GF}(q)$, *channel* alphabet).

The *error polynomial* is

$$e(x) = e_{i_1} x^{i_1} + \cdots + e_{i_\nu} x^{i_\nu} .$$

The senseword $r(x)$ can be written $r(x) = c(x) + e(x)$.

The partial syndromes are values in *decoder* alphabet $\mathrm{GF}(q^m)$ :

$$\begin{aligned} S_j = r(\alpha^j) \;&=\; c(\alpha^j) + e(\alpha^j) \;=\; e(\alpha^j) \\ &= e_{i_1}\alpha^{ji_1} + \cdots + e_{i_\nu}\alpha^{ji_\nu} \;=\; e_{i_1}\alpha^{i_1 j} + \cdots + e_{i_\nu}\alpha^{i_\nu j} . \end{aligned}$$

Change of variables:

- *error locators*: $X_1 = \alpha^{i_1}, \ldots, X_\nu = \alpha^{i_\nu}$
- *error magnitudes*: $Y_1 = e_{i_1}, \ldots, Y_\nu = e_{i_\nu}$ (just renaming)

# Syndrome equations

Error locators are elements of the *decoder* alphabet $\mathrm{GF}(q^m)$.

Error magnitudes are elements of the *channel* alphabet $\mathrm{GF}(q)$.

Important special case: $Y_i = 1$ for channel alphabet $\mathrm{GF}(2)$.

For now, assume narrow-sense BCH code $(b = 1)$ with $d = 2t + 1$.

Partial syndromes are constants in system of $2t$ equations in $2\nu$ unknowns:

$$\begin{aligned} S_1 &= Y_1 X_1 + \cdots + Y_\nu X_\nu \\ S_2 &= Y_1 X_1^2 + \cdots + Y_\nu X_\nu^2 \\ &\vdots \\ S_{2t} &= Y_1 X_1^{2t} + \cdots + Y_\nu X_\nu^{2t} \end{aligned}$$

This is an algebraic system of equations of degree $2t$.

Goal: reduce to one-variable polynomial equation with $\nu$ solutions.

# Error-locator polynomial

The *error-locator polynomial* $\Lambda(x)$ is defined by

$$\begin{aligned}\Lambda(x) &= (1 - xX_1)(1 - xX_2)\cdots(1 - xX_\nu) \\ &= \prod_{i=1}^{\nu}(1 - xX_i) \\ &= \prod_{i=1}^{\nu}(-X_i) \cdot \prod_{i=1}^{\nu}(x - X_i^{-1}) \\ &= 1 + \Lambda_1 x + \cdots + \Lambda_\nu x^\nu \,.\end{aligned}$$

The zeroes of $\Lambda(x)$ are $X_1^{-1}, \ldots, X_\nu^{-1}$: the *reciprocals* of error locators.

The degree of $\Lambda(x)$ is the number of errors. elegant!

The decoder must determine $\nu$ as well as the error locations.

The Peterson-Gorenstein-Zierler decoder can be used to find $\Lambda(x)$ from $S_j$.

PGZ is not efficient for large $t$ but is easy to understand.

# PGZ decoder example

2EC narrow-sense BCH code with decoder alphabet $\mathrm{GF}(2^m)$ syndromes:

$$S_j = Y_1 X_1^j + Y_2 X_2^j\,, \quad j = 1, \ldots, 4$$

Suppose two errors. Then zeroes of $\Lambda(x) = 1 + \Lambda_1 x + \Lambda_2 x^2$ are $X_1^{-1}, X_2^{-1}$.

$$0 = 1 + \Lambda_1 X_1^{-1} + \Lambda_2 X_1^{-2} \xrightarrow{\cdot\, Y_1 X_1^3} Y_1 X_1^3 + \Lambda_1 Y_1 X_1^2 + \Lambda_2 Y_1 X_1 = 0$$

$$0 = 1 + \Lambda_1 X_2^{-1} + \Lambda_2 X_2^{-2} \xrightarrow{\cdot\, Y_2 X_2^3} Y_2 X_2^3 + \Lambda_1 Y_2 X_2^2 + \Lambda_2 Y_2 X_2 = 0$$

$$\underbrace{(Y_1 X_1^3 + Y_2 X_2^3)}_{S_3} + \Lambda_1 \underbrace{(Y_1 X_1^2 + Y_2 X_2^2)}_{S_2} + \Lambda_2 \underbrace{(Y_1 X_1 + Y_2 X_2)}_{S_1} = 0$$

Similarly, multiplying by $Y_i X_i^4$ and summing gives another equation:

$$\underbrace{(Y_1 X_1^4 + Y_2 X_2^4)}_{S_4} + \Lambda_1 \underbrace{(Y_1 X_1^3 + Y_2 X_2^3)}_{S_3} + \Lambda_2 \underbrace{(Y_1 X_1^2 + Y_2 X_2^2)}_{S_2} = 0$$

We have obtained two linear equations in the unknowns $\Lambda_1, \Lambda_2$:

$$\begin{aligned} S_3 + S_2\Lambda_1 + S_1\Lambda_2 &= 0 \\ S_4 + S_3\Lambda_1 + S_3\Lambda_2 &= 0 \end{aligned} \quad \Longrightarrow \quad \begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \begin{bmatrix} \Lambda_2 \\ \Lambda_1 \end{bmatrix} = -\begin{bmatrix} S_3 \\ S_4 \end{bmatrix}.$$

# PGZ decoder example (cont.)

The determinant of the coefficient matrix is

$$\Delta = S_1 S_3 - S_2^2 = (Y_1 X_1 + Y_2 X_2)(Y_1 X_1^3 + Y_2 X_2^3) + (Y_1 X_1^2 + Y_2 X_2^2)^2$$
$$= Y_1 Y_2 (X_1 X_2^3 + X_1^3 X_2) = Y_1 Y_2 X_1 X_2 (X_1 + X_2)^2$$

$\Delta \neq 0$ because $Y_i \neq 0$, $X_i \neq 0$, $X_1 \neq X_2$. So we can solve for $\Lambda_1, \Lambda_2$.

$$M_2 = \begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \implies M_2^{-1} = \Delta^{-1} \begin{bmatrix} S_3 & S_2 \\ S_2 & S_1 \end{bmatrix}$$

where $\Delta = \det M_2 = S_1 S_3 + S_2^2$. Coefficients of $\Lambda(x)$ are given by

$$\begin{bmatrix} \Lambda_2 \\ \Lambda_1 \end{bmatrix} = \Delta^{-1} \begin{bmatrix} S_3 & S_2 \\ S_2 & S_1 \end{bmatrix} \begin{bmatrix} S_3 \\ S_4 \end{bmatrix} = \Delta^{-1} \begin{bmatrix} S_3^2 + S_2 S_4 \\ S_2 S_3 + S_1 S_4 \end{bmatrix}$$

The error locator polynomial is $\Lambda(x) = 1 + \Lambda_1 x + \Lambda_2 x^2$, where

$$\Lambda_1 = \frac{S_2 S_3 + S_1 S_4}{S_1 S_3 + S_2^2}, \quad \Lambda_2 = \frac{S_3^2 + S_2 S_4}{S_1 S_3 + S_2^2}.$$

The common denominator $\Delta = S_1 S_3 + S_2^2$ need be computed only once.

Computation of $\Lambda_1, \Lambda_2$ uses 8 multiplications and one inversion in $\mathrm{GF}(2^m)$.

# PGZ decoder example (cont.)

Next find two zeroes $X_1^{-1}, X_2^{-1}$ of $\Lambda(x)$ (perhaps by exhaustive search).

If $\Lambda(x)$ does not have two distinct zeroes, an uncorrectable error has occurred.

Finally find the error magnitudes:

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 \\ X_1^2 & X_2^2 \end{bmatrix}^{-1} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = \frac{1}{X_1 X_2 (X_1 + X_2)} \begin{bmatrix} X_2^2 & X_2 \\ X_1^2 & X_1 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}$$

Matrix-vector product gives error magnitudes:

$$Y_1 = \frac{X_2^2 S_1 + X_2 S_2}{X_1 X_2 (X_1 + X_2)} = \frac{X_2 S_1 + S_2}{X_1 (X_1 + X_2)}$$
$$Y_2 = \frac{X_1^2 S_1 + X_1 S_2}{X_1 X_2 (X_1 + X_2)} = \frac{X_1 S_1 + S_2}{X_2 (X_1 + X_2)}$$

Computation of $Y_1, Y_2$ takes about 6 multiplications and 2 reciprocals.

# PGZ decoder example (cont.)

If $M_2$ is singular, that is, $S_1 S_3 + S_2^2 = 0$, then we solve the simpler equation

$$M_1 [\, \Lambda_1 \,] = [\, S_2 \,] \implies [\, S_1 \,][\, \Lambda_1 \,] = [\, S_2 \,]$$

The error locator polynomial has degree $1$:

$$\Lambda(x) = 1 + \Lambda_1 x = 1 + \frac{S_2}{S_1} x \implies X_1^{-1} = \frac{S_1}{S_2}$$

If $S_2 \neq 0$ then the single error locator is the reciprocal of the zero of $\Lambda(x)$:

$$X_1 = \frac{S_2}{S_1} = \frac{Y_1 X_1^2}{Y_1 X_1}$$

Error magnitude is obtained from $S_1 = Y_1 X_1$:

$$Y_1 = \frac{S_1}{X_1} = \frac{S_1^2}{S_2} = \frac{Y_1^2 X_1^2}{Y_1 X_1^2} \,.$$

Finally we check $S_4 = Y_1 X_1^4$. If not, an uncorrectable error has been detected.

# PGZ in general

By definition of the error locator polynomial, $\Lambda(X_i^{-1}) = 0$, i.e.,

$$1 + \Lambda_1 X_i^{-1} + \cdots + \Lambda_\nu X_i^{-\nu} = 0 \qquad (i = 1, \ldots, \nu)$$

Multiply this equation by $Y_i X_i^{j+\nu}$ for any $j \geq 1$:

$$Y_i X_i^{j+\nu} + \Lambda_1 Y_i X_i^{j+\nu-1} + \cdots + \Lambda_\nu Y_i X_i^{j} = 0$$

This equation has only positive powers of $X_i$. Now sum over $i$:

$$\sum_{i=1}^{\nu} Y_i X_i^{j+\nu} + \Lambda_1 \sum_{i=1}^{\nu} Y_i X_i^{j+\nu-1} + \cdots + \Lambda_\nu \sum_{i=1}^{\nu} Y_i X_i^{j} = 0$$

Iff $j \geq 1$ and $j + \nu \leq 2t$, that is, $1 \leq j \leq 2t - \nu$,

$$S_{j+\nu} + \Lambda_1 S_{j+\nu-1} + \cdots + \Lambda_\nu S_j = 0$$

We have obtained $2t - \nu \geq \nu$ linear equations in $\nu$ unknowns $\Lambda_1, \ldots, \Lambda_\nu$:

$$S_j \Lambda_\nu + S_{j+1} \Lambda_{\nu-1} + \cdots + S_{j+\nu-1} \Lambda_1 = -S_{j+\nu}$$

# Linear equations for $\Lambda_1, \ldots, \Lambda_\nu$

The first $\nu$ linear equations for $\Lambda_1, \ldots, \Lambda_\nu$ have a $\nu \times \nu$ coefficient matrix:

$$\begin{bmatrix} S_1 & S_2 & \cdots & S_\nu \\ S_2 & S_3 & \cdots & S_{\nu+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_\nu & S_{\nu+1} & \cdots & S_{2\nu-1} \end{bmatrix} \begin{bmatrix} \Lambda_\nu \\ \Lambda_{\nu-1} \\ \vdots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -S_{\nu+1} \\ -S_{\nu+2} \\ \vdots \\ -S_{2\nu} \end{bmatrix}$$

For any $\mu = 1, 2, \ldots, t$, let $M_\mu$ be the matrix

$$M_\mu = \begin{bmatrix} S_1 & S_2 & \cdots & S_\mu \\ S_2 & S_3 & \cdots & S_{\mu+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_\mu & S_{\mu+1} & \cdots & S_{2\mu-1} \end{bmatrix}.$$

*Lemma*: Suppose that there are $\nu \leq t$ symbol errors. Then $M_\nu$ is nonsingular, but $M_\mu$ is singular for $\mu > \nu$.

Matrices that are constant along anti-diagonals are called *Hankel* matrices.

# Determining number of errors $\nu$

*Proof*: Let $X_i = 0$ when $\nu < i \leq t$. Syndrome equations hold.

$$M_\mu = \begin{bmatrix} S_1 & \cdots & S_\mu \\ \vdots & \ddots & \vdots \\ S_\mu & \cdots & S_{2\mu-1} \end{bmatrix} = \begin{bmatrix} \sum_1^\mu Y_i X_i^1 & \cdots & \sum_1^\mu Y_i X_i^\mu \\ \sum_1^\mu Y_i X_i^2 & \cdots & \sum_1^\mu Y_i X_i^{\mu+1} \\ \vdots & \ddots & \vdots \\ \sum_1^\mu Y_i X_i^\mu & \cdots & \sum_1^\mu Y_i X_i^{2\mu-1} \end{bmatrix}$$

$$= \begin{bmatrix} Y_1 X_1 & \cdots & Y_\mu X_\mu \\ \vdots & \ddots & \vdots \\ Y_1 X_1^\mu & \cdots & Y_\mu X_\mu^\mu \end{bmatrix} \cdot \begin{bmatrix} 1 & \cdots & 1 \\ X_1 & \cdots & X_\mu \\ \vdots & \ddots & \vdots \\ X_1^{\mu-1} & \cdots & X_\mu^{\mu-1} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ X_1^{\mu-1} & \cdots & X_\mu^{\mu-1} \end{bmatrix} \cdot \begin{bmatrix} Y_1 X_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & Y_\mu X_\mu \end{bmatrix} \cdot \begin{bmatrix} 1 & \cdots & X_1^{\mu-1} \\ \vdots & \ddots & \vdots \\ 1 & \cdots & X_\mu^{\mu-1} \end{bmatrix}$$

If $i \leq \nu$ then $X_i \neq 0$ and $Y_i \neq 0$. Therefore $M_\nu$ is the product of nonsingular Vandermonde and diagonal matrices and is nonsingular.

But if $\mu > \nu$ the middle matrix has a zero element $Y_\mu X_\mu$ on its diagonal. The middle matrix is singular for $\mu > \nu$ and therefore $M_\mu$ is singular.

# Peterson-Gorenstein-Zierler (PGZ) decoder: summary

1. Compute partial syndromes $S_j = r(\alpha^j)$.
2. Find largest $\nu \leq t$ such that $\det M_\nu \neq 0$.
3. Solve the following *linear* system for the coefficients of $\Lambda(x)$.
$$M_\nu \left[ \Lambda_\nu, \ldots, \Lambda_1 \right]^T = \left[ -S_{\nu+1}, \ldots, -S_{2\nu} \right]^T$$
4. Find $X_1^{-1}, \ldots, X_\nu^{-1}$, zeroes of $\Lambda(x)$ in $\mathrm{GF}(q^m)$, decoder alphabet. If $\Lambda(x)$ has $< \nu$ distinct zeroes, an uncorrectable error has occurred.
5. Solve following system of linear equations for error magnitudes.
$$\begin{array}{rcl} Y_1 X_1 + \cdots + Y_\nu X_\nu & = & S_1 \\ Y_1 X_1^2 + \cdots + Y_\nu X_\nu^2 & = & S_2 \\ \vdots & & \vdots \\ Y_1 X_1^\nu + \cdots + Y_\nu X_\nu^\nu & = & S_\nu \\ \vdots & & \vdots \\ Y_1 X_1^{2t} + \cdots + Y_\nu X_\nu^{2t} & = & S_{2t} \end{array}$$

The Forney algorithm, $Y_i = -\frac{\Omega(X_i^{-1})}{\Lambda'(X_i^{-1})}$, is a "closed" form solution for step 5.

# 3EC Reed-Solomon code

Narrow-sense BCH codes usually do not have the simplest generator polynomial or parity-check matrices.

For that reason, Reed-Solomon codes are usually defined using $b = 0$.

The following matrix defines a three error correcting Reed-Solomon code:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \cdots & \alpha^{2(n-1)} \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 & \cdots & \alpha^{3(n-1)} \\ 1 & \alpha^4 & \alpha^8 & \alpha^{12} & \cdots & \alpha^{4(n-1)} \\ 1 & \alpha^5 & \alpha^{10} & \alpha^{15} & \cdots & \alpha^{5(n-1)} \end{bmatrix}$$

The generator polynomial is

$$g(x) = (x+1)(x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4)(x+\alpha^5)$$

Another trick to reduce encoder complexity is to choose $b$ so that the generator polynomial is reversible — inverses of zeroes are also zeroes — so half as many scalers are needed in the encoding and syndrome circuits.

# 3EC Reed-Solomon decoding (cont.)

Partial syndromes defined by $S_j = r(\alpha^j)$ for $j = 0, \ldots, 5$ satisfy equations

$$\begin{aligned}
S_0 &= Y_1 + Y_2 + Y_3 \\
S_1 &= Y_1X_1 + Y_2X_2 + Y_3X_3 \\
S_2 &= Y_1X_1^2 + Y_2X_2^2 + Y_3X_3^2 \\
S_3 &= Y_1X_1^3 + Y_2X_2^3 + Y_3X_3^3 \\
S_4 &= Y_1X_1^4 + Y_2X_2^4 + Y_3X_3^4 \\
S_5 &= Y_1X_1^5 + Y_2X_2^5 + Y_3X_3^5
\end{aligned}$$

where $X_1, X_2, X_3$ are error location numbers and $Y_1, Y_2, Y_3$ are error magnitudes.

The coefficients of the error locator polynomial $\Lambda(x)$ satisfy the linear equations:

$$\begin{bmatrix} S_0 & S_1 & S_2 \\ S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \end{bmatrix} \begin{bmatrix} \Lambda_3 \\ \Lambda_2 \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} S_3 \\ S_4 \\ S_5 \end{bmatrix}$$

# 3EC Reed-Solomon decoding (cont.)

If there are three errors, then the solutions can be found using Cramer's rule:

$$\Lambda_0 = S_2(S_1S_3 + S_2S_2) + S_3(S_0S_3 + S_1S_2) + S_4(S_0S_2 + S_1S_1)$$
$$\Lambda_1 = S_3(S_1S_3 + S_2S_2) + S_4(S_0S_3 + S_1S_2) + S_5(S_0S_2 + S_1S_1)$$
$$\Lambda_2 = S_3(S_1S_4 + S_2S_3) + S_4(S_0S_4 + S_2S_2) + S_5(S_0S_3 + S_1S_2)$$
$$\Lambda_3 = S_3(S_2S_4 + S_3S_3) + S_4(S_1S_4 + S_2S_3) + S_5(S_1S_3 + S_2S_2)$$

Note that we can choose $\Lambda_0 = 1$ by dividing the other coefficients by $\Lambda_0$.

Let $X_1, X_2, X_3$ be the zeroes of

$$\Lambda(x) = \Lambda_0 + \Lambda_1 x + \Lambda_2 x^2 + \Lambda_3 x^3 \, .$$

The location of the incorrect symbols are $i_1, i_2, i_3$, where

$$X_1 = \alpha^{i_1} \, , \quad X_2 = \alpha^{i_2} \, , \quad X_3 = \alpha^{i_3} \, .$$

# 3EC Reed-Solomon decoding (cont.)

Finally, the error magnitudes $Y_1, Y_2, Y_3$ can be found by solving equations that use the first three syndrome components, $S_0, S_1, S_2$:

$$Y_1 = \frac{S_2 + S_1(X_2 + X_3) + S_0 X_2 X_3}{(X_1 + X_2)(X_1 + X_3)}$$

$$Y_2 = \frac{S_2 + S_1(X_1 + X_3) + S_0 X_1 X_3}{(X_2 + X_1)(X_2 + X_3)}$$

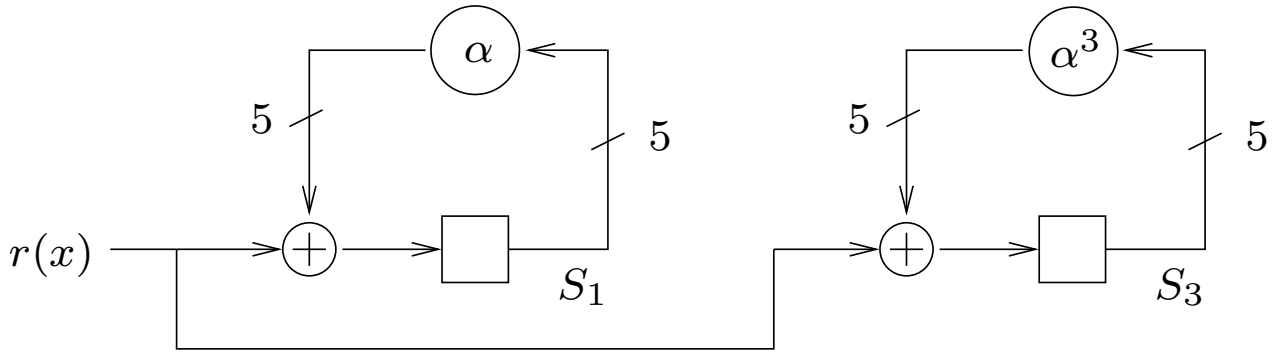
$$Y_3 = \frac{S_2 + S_1(X_1 + X_2) + S_0 X_1 X_2}{(X_3 + X_1)(X_3 + X_2)}$$

Starting from the partial syndromes $S_0, S_1, \ldots, S_5$, approximately 30 Galois field multiplications and 3 Galois field divisions are needed for decoding.

This estimate does not count the effort needed to find the zeroes of $\Lambda(x)$.

# Partial syndrome circuits for GF(32)

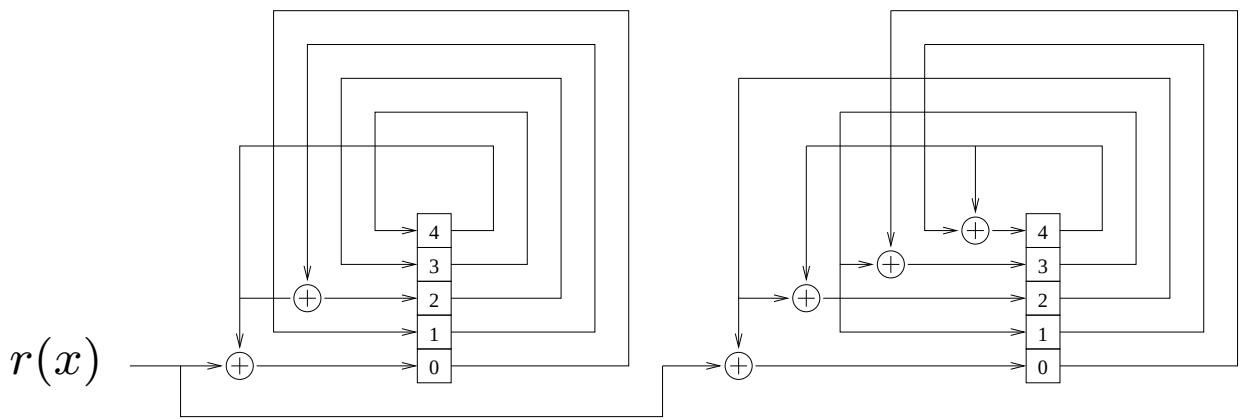
Horner's method for polynomial evaluation:



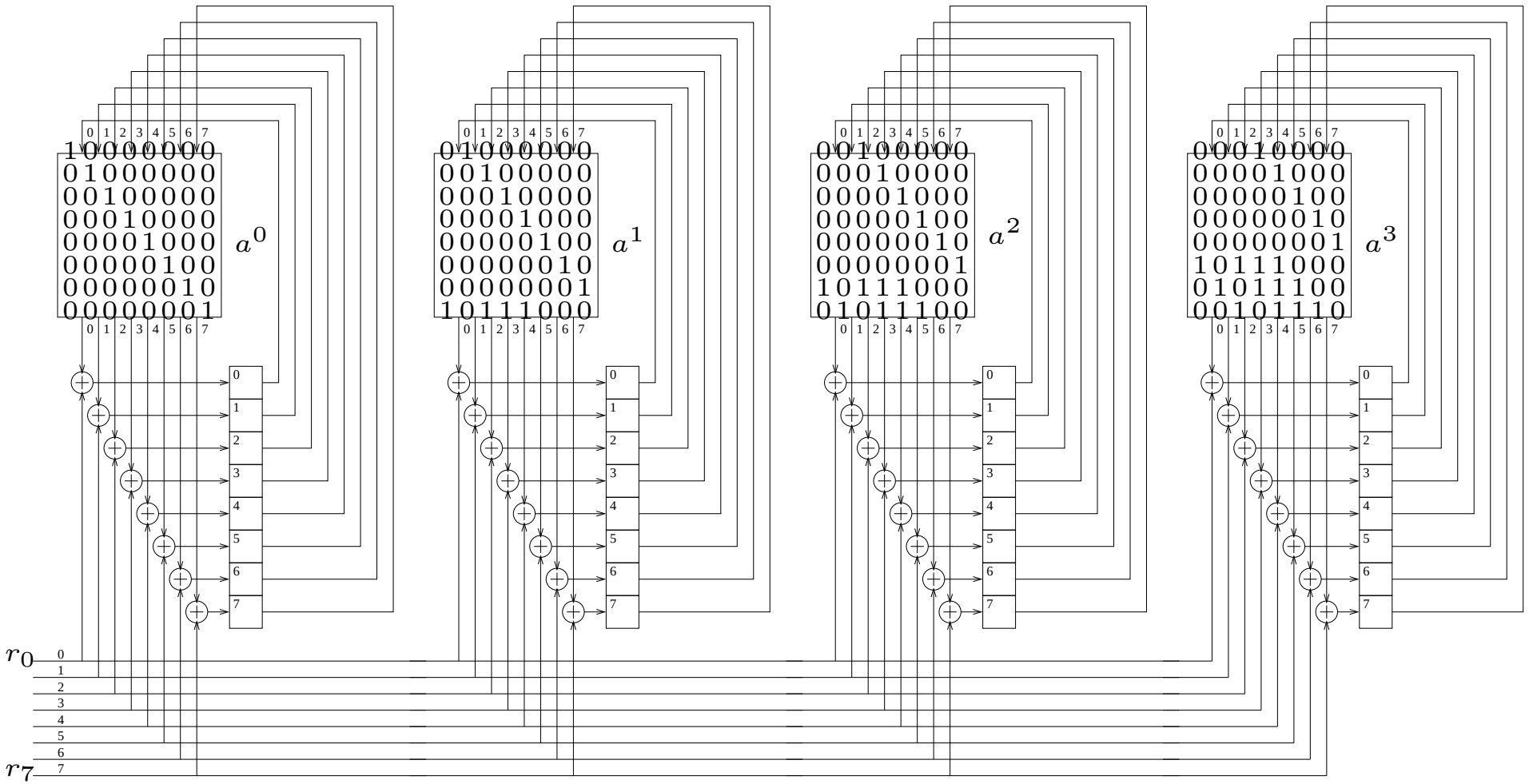

Multiplication by $\alpha$, $\alpha^3$ uses matrices: $(\alpha^5 + \alpha^2 + 1 = 0)$

$$M_\alpha = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad M_{\alpha^3} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Circuit for $r(\alpha)$ and $r(\alpha^3)$



# Partial syndrome circuit for $(255, 251)$ R-S code



# Chien search

The Chien search is a clever method for finding zeroes of the error locator polynomial by brute force.

The Chien search evaluates $\Lambda(\alpha^i)$ for $i = 1, 2, \ldots, n$ using $\nu$ *constant* multiplications instead of $\nu$ general multiplications.

Key idea: use $\nu$ state variables $Q_1, \ldots, Q_\nu$ such that at time $i$

$$Q_j = \Lambda_j \alpha^{ji}\,, \quad j = 1, \ldots, \nu\,.$$

Each state variable is updated by multiplication by a constant:

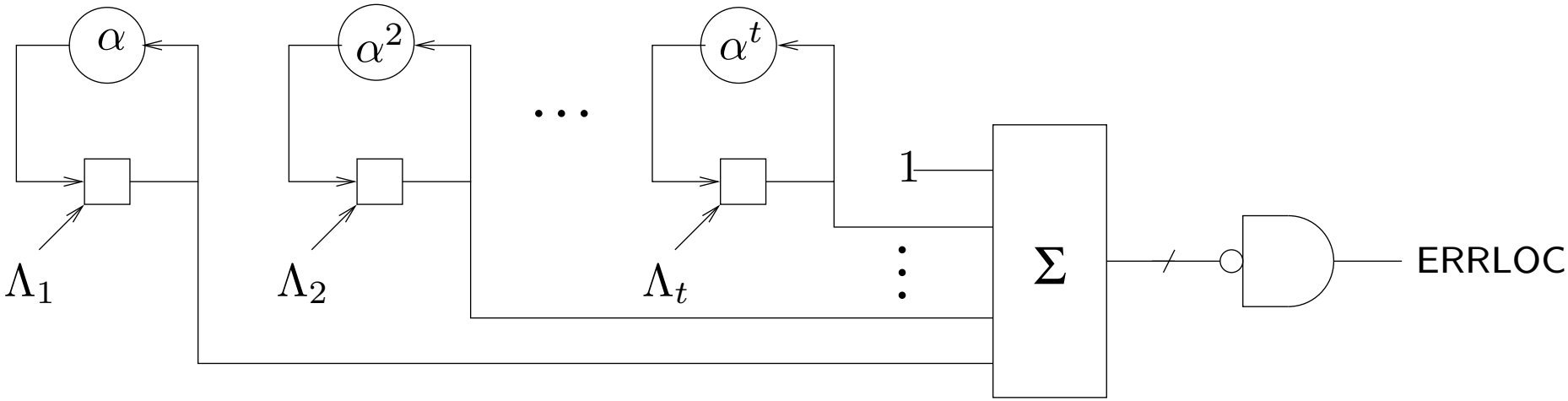
$$Q_j \to Q_j \alpha^j\,, \quad i = 1, \ldots, n\,.$$

Sum of state variables at time $i$ is $\displaystyle\sum_{j=1}^{\nu} Q_j = \Lambda(\alpha^i) - 1\,.$

An error location is identified whenever this sum equals $-1$.

# Chien search circuit #1

Memory elements are initialized with coefficients of error locator polynomial where we set $\Lambda_j = 0$ for $j = \nu + 1, \ldots, t$.




Output signal ERRLOC is true when $\Lambda(\alpha^i) = 0$.

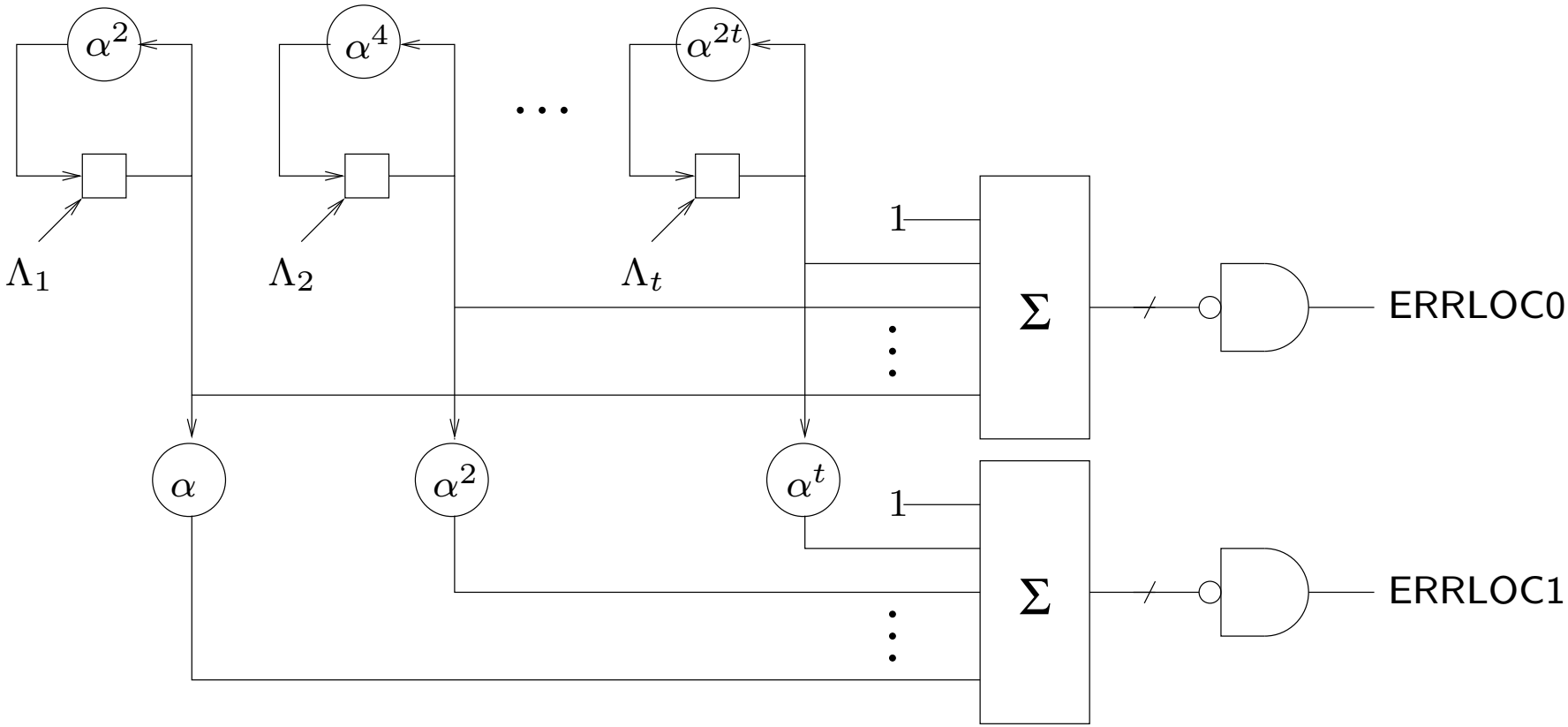
Since the zeroes of $\Lambda(x)$ are the reciprocals of the error location numbers, ERRLOC is true for values of $i$ such that $\alpha^{-i} = \alpha^{n-i} = X_l$.

As $i$ runs from 1 to $n$, error locations are detected from msb down to lsb.

Chien search can also be run backwards, using scalers for $\alpha^{-1}, \ldots, \alpha^{-t}$.

# Chien search circuit #2

Double-speed Chien search: evaluate $\Lambda(\alpha^{2i})$ and $\Lambda(\alpha^{2i+1})$ at same time.



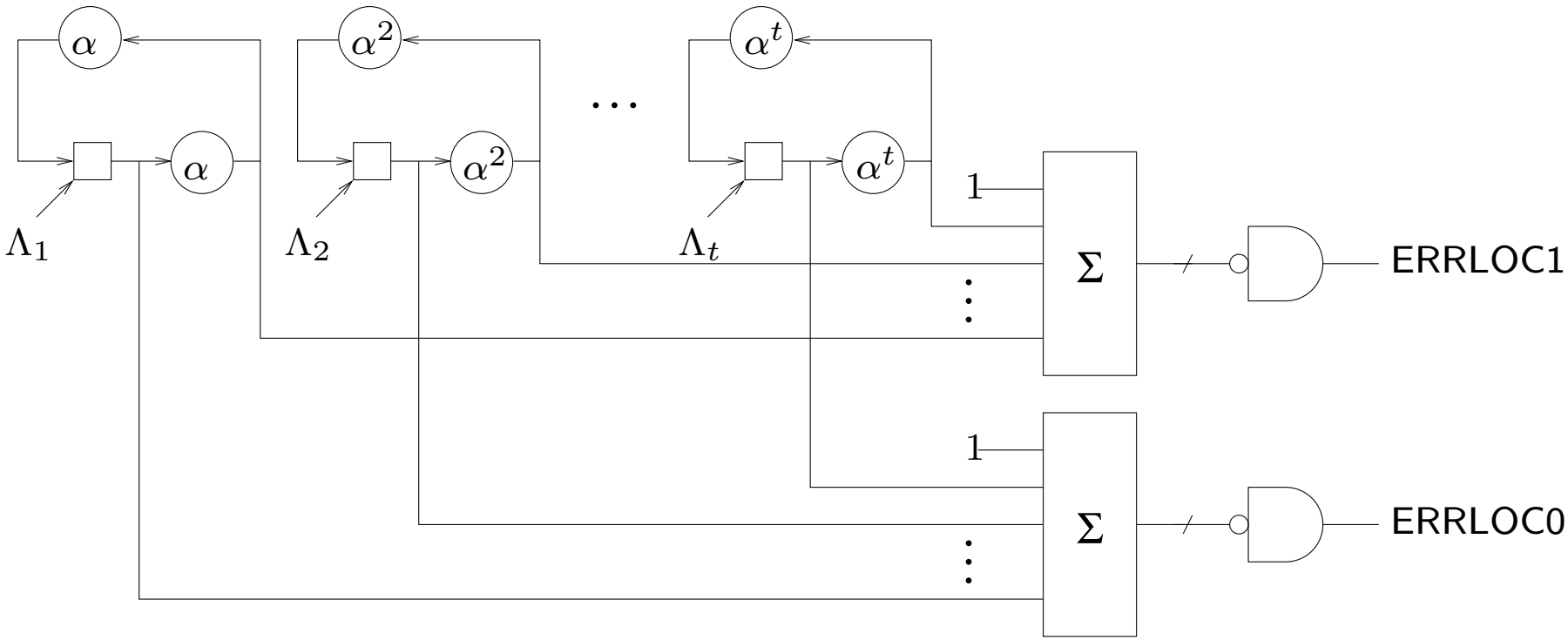

ERRLOC0 (ERRLOC1) is asserted when even (odd) error location is found.

This circuit is more efficient than two separate copies of the Chien search engine because the memory storage elements are shared.

# Chien search circuit #3

Scaler $\alpha^{2j}$ usually requires more gates than scaler $\alpha^j$ for small values of $j$.

We can reduce cost of double-speed Chien search by using $\alpha^{2j} = \alpha^j \cdot \alpha^j$




Cascade of two scalers for $\alpha^i$ may be slightly slower than one scaler for $\alpha^{2i}$.

# PGZ decoder: review

1. Compute partial syndromes $S_j = r(\alpha^j)$.
2. Solve a linear system of equations for coefficients of $\Lambda(x)$:

   $$M_\nu \left[ \Lambda_\nu, \ldots, \Lambda_1 \right]^T = \left[ -S_{\nu+1}, \ldots, -S_{2\nu} \right]^T$$

   where $\nu$ is the largest number $\leq t$ such that $\det M_\nu \neq 0$.
3. Find the zeroes of $\Lambda(x)$ are $X_1^{-1}, \ldots, X_\nu^{-1}$, which are the reciprocals of the error locators $X_1 = \alpha^{i_1}, \ldots, X_\nu = \alpha^{i_\nu}$.
4. Solve a system of linear equations for the error magnitudes $Y_1, \ldots, Y_\nu$.

$$\begin{array}{rcccl} Y_1 X_1 & + \cdots + & Y_\nu X_\nu & = & S_1 \\ Y_1 X_1^2 & + \cdots + & Y_\nu X_\nu^2 & = & S_2 \\ & \vdots & & & \vdots \\ Y_1 X_1^{2t} & + \cdots + & Y_\nu X_\nu^{2t} & = & S_{2t} \end{array}$$

The Forney algorithm (1965) is a simple closed formula for $Y_1, \ldots, Y_\nu$.

# Forney Algorithm

Consider a BCH code defined by the zeroes $\alpha^b, \alpha^{b+1}, \ldots, \alpha^{b+2t-1}$.

Forney algorithm: error magnitude $Y_i$ corresponding to error locator $X_i$ is

$$Y_i = -\frac{X_i^{1-b}\Omega(X_i^{-1})}{\Lambda'(X_i^{-1})}$$

where $\Lambda'(x)$ is the *formal derivative* of the error-locator polynomial:

$$\Lambda'(x) = \sum_{i=1}^{\nu} i\Lambda_i x^{i-1}$$

and $\Omega(x)$ is the *error evaluator* polynomial:

$$S(x)\Lambda(x) \bmod x^{2t}$$

The Forney algorithm simplifies for narrow-sense BCH codes ($b = 1$):

$$Y_i = -\frac{\Omega(X_i^{-1})}{\Lambda'(X_i^{-1})}.$$

Forney's algorithm uses $2\nu^2$ multiplications to compute all error magnitudes.

# Partial syndrome polynomial

*Definition*: The *partial syndrome polynomial* for a narrow-sense BCH code is the *generating function* of the sequence $S_1, S_2, \ldots, S_{2t}$ :

$$S(x) = S_1 + S_2 x + S_3 x^2 + \cdots + S_{2t} x^{2t-1} .$$

For BCH code defined by $\alpha^b, \ldots, \alpha^{b+2t-1}$, partial syndrome polynomial is

$$S(x) = S_b + S_{b+1} x + \cdots + S_{b+2t-1} x^{2t-1} .$$

The PGZ decoder solves linear equations for coefficients of $\Lambda(x)$.
For $j = 1, \ldots, 2t - \nu$:

$$S_j \Lambda_\nu + \cdots + S_{j+\nu-1} \Lambda_1 + S_{j+\nu} = 0 \qquad \text{(narrow sense codes)}$$

$$S_{b+j-1} \Lambda_\nu + \cdots + S_{b+j+\nu-2} \Lambda_1 + S_{b+j+\nu-1} = 0 \qquad \text{(general BCH codes)}$$

In either case, the left hand side is the coefficient of $x^{\nu+j-1}$ in the polynomial product $S(x)\Lambda(x)$.

We can define partial syndromes $S_i$ for all $i > 0$, but decoder can compute only first $2t$ values.

# Error evaluator polynomial

The *error evaluator polynomial* $\Omega(x)$ is defined by the *key equation*:

$$\Omega(x) \;=\; S(x)\Lambda(x) \bmod x^{2t}\,,$$

where $S(x)$ is partial syndrome polynomial and $\Lambda(x)$ is error-locator polynomial.

If $1 \le j \le 2t-\nu$ then by PGZ equations the coefficient of $x^{\nu+j-1}$ in the product $S(x)\Lambda(x)$ is $0$.

Therefore $\deg\left(S(x)\Lambda(x) \bmod x^{2t}\right) < \nu$ if there are $\nu \le t$ errors.

The error evaluator polynomial can be computed explicitly from $\Lambda(x)$:

$$\begin{aligned}
\Omega_0 &= S_b \\
\Omega_1 &= S_{b+1} + S_b\Lambda_1 \\
\Omega_2 &= S_{b+2} + S_{b+1}\Lambda_1 + S_b\Lambda_2 \\
&\vdots \\
\Omega_{\nu-1} &= S_{b+\nu-1} + S_{b+\nu-2}\Lambda_1 + \cdots + S_b\Lambda_{\nu-1}
\end{aligned}$$

Multiply-accumulates needed: $0+1+\cdots+\nu-2 = \frac{1}{2}(\nu-1)(\nu-2) \approx \frac{1}{2}\nu^2$

# Formal derivative

The Forney algorithm is a closed formula for $Y_i$ in terms of $\Omega(x)$, $\Lambda(x)$, and $X_i$.

First we need the notion of the formal derivative of a polynomial.

*Definition*: The *formal derivative* of

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_n x^n$$

is the polynomial

$$f'(x) = f_1 + 2f_2 x + \cdots + k f_k x^{k-1} + \cdots + n f_n x^{n-1}$$

where $k$ is the sum of $k$ 1s and is a field integer.

Most familiar properties of derivatives hold. In particular, product rules:

$$(f(x)g(x))' = f'(x)g(x) + f(x)g'(x)$$

$$\Big(\prod_{i=1}^{n} f_i(x)\Big)' = \sum_{i=1}^{n} \Big(f_i'(x) \prod_{j \neq i} f_j(x)\Big)$$

Formal derivatives of polynomials are defined algebraically, not by taking limits.

# Properties of formal derivatives

*Fact*: A polynomial $f(x)$ over $\mathrm{GF}(q)$ has a repeated zero $\beta$ iff $f'(\beta) = 0$.

*Proof*: If $\beta$ is a zero of $f(x)$, then $x - \beta$ is a factor of $f(x)$:

$$f(x) = f_1(x)(x - \beta)$$

Therefore $f'(x) = f_1'(x)(x - \beta) + f_1(x) \implies f'(\beta) = f_1(\beta)$.

Thus $\beta$ is a repeated zero — $f(x)$ has factor $(x - \beta)^2$ — iff $f'(\beta) = 0$.

Over $\mathrm{GF}(2^m)$, formal derivative has only even powers of the indeterminant:

$$\begin{aligned} f'(x) &= f_1 + 2f_2x + 3f_3x^2 + \cdots + nf_nx^{n-1} \\ &= f_1 + 3f_3x^2 + 5f_5x^4 + \cdots \end{aligned}$$

since $2 = 1 + 1 = 0$, $4 = 2 + 2 = 2(1 + 1) = 0$, and so on. So the formal derivative $\Lambda'(x)$ has at most $\nu/2$ nonzero coefficients.

Since $\Lambda'(x)$ is a polynomial in $x^2$ of degree $< \nu/2$, we can compute $\Lambda'(\beta)$ using one squaring and $\leq \nu/2$ multiply-accumulate operations.

Note that $f''(x) = 0$ for all polynomials over $\mathrm{GF}(2^m)$.

# Forney algorithm: derivation

We can express error evaluator $\Omega(x)$ in terms of error locator $X_i$ and error magnitudes $Y_i$.

First we derive a closed formula for $S(x)$.

$$
\begin{aligned}
S(x) &= \sum_{j=0}^{2t-1} S_{b+j} x^j \\
&= \sum_{j=0}^{2t-1} \sum_{i=1}^{\nu} Y_i X_i^{b+j} x^j \\
&= \sum_{i=1}^{\nu} \sum_{j=0}^{2t-1} Y_i X_i^{b+j} x^j \\
&= \sum_{i=1}^{\nu} Y_i X_i^b \sum_{j=0}^{2t-1} X_i^j x^j = \sum_{i=1}^{\nu} Y_i X_i^b \, \frac{1-(X_i x)^{2t}}{1-X_i x}
\end{aligned}
$$

Next we use the definition $\Lambda(x) = \prod_{l=1}^{\nu} \left(1 - X_l x\right)$ to compute $S(x)\Lambda(x)$.

# Forney algorithm: derivation (cont.)

$$
\begin{aligned}
S(x)\Lambda(x) &= \sum_{i=1}^{\nu} \left( Y_i X_i^b \, \frac{1-(X_i x)^{2t}}{1-X_i x} \right) \cdot \prod_{l=1}^{\nu} (1 - X_l x) \\
&= \sum_{i=1}^{\nu} \left( Y_i X_i^b \prod_{l \neq i} (1 - X_l x) \big(1 - (X_i x)^{2t}\big) \right) \\
&= \sum_{i=1}^{\nu} Y_i X_i^b \prod_{l \neq i} (1 - X_l x) \; - \; \sum_{i=1}^{\nu} Y_i X_i^b (X_i x)^{2t} \prod_{l \neq i} (1 - X_l x)
\end{aligned}
$$

The second sum in the final expression is a polynomial in $x$ of degree $\geq 2t$.

Thus the remainder modulo $x^{2t}$ of the second sum is $0$.

Therefore

$$
\Omega(x) = S(x)\Lambda(x) \bmod x^{2t} = \sum_{i=1}^{\nu} Y_i X_i^b \prod_{l \neq i} (1 - X_l x) \, .
$$

# Forney algorithm: derivation (cont.)

Now we use the product formula for $\Lambda'(x)$:

$$\Lambda'(x) = \left( \prod_{l=1}^{\nu} (1 - X_l x) \right)' = \sum_{l=1}^{\nu} (-X_l) \prod_{j \neq l} (1 - X_j x)$$

When we evaluate $\Lambda'(x)$ at $X_i^{-1}$, only one term in the sum is nonzero:

$$\Lambda'(X_i^{-1}) = -X_i \prod_{j \neq i} (1 - X_j X_i^{-1})$$

Similarly, the value of $\Omega(x)$ at $X_i^{-1}$ includes only one term from the sum:

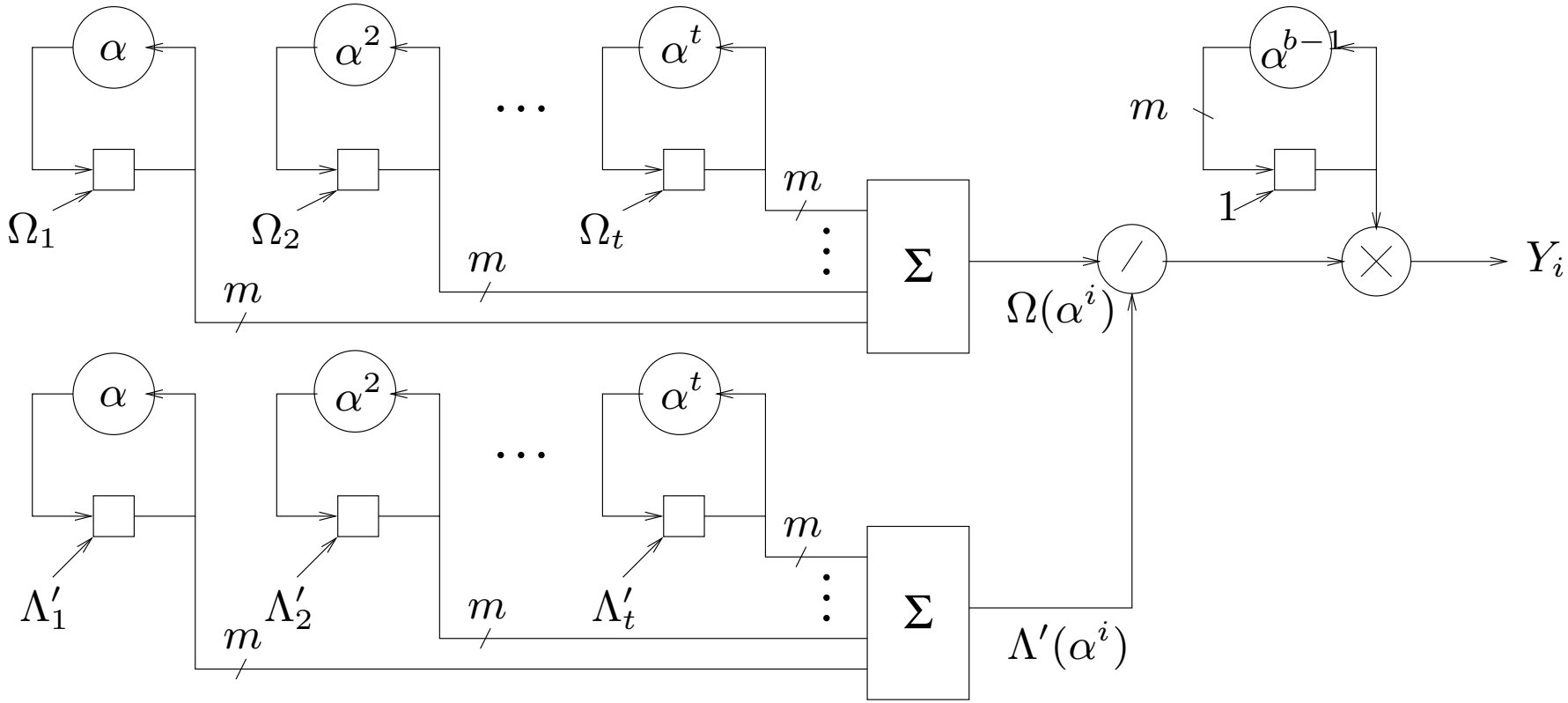
$$\Omega(X_i^{-1}) = \sum_{l=1}^{\nu} Y_l X_l^b \prod_{j \neq l} (1 - X_j X_i^{-1}) = Y_i X_i^b \prod_{j \neq i} (1 - X_j X_i^{-1})$$

Thus

$$\frac{\Omega(X_i^{-1})}{\Lambda'(X_i^{-1})} = \frac{Y_i X_i^b}{-X_i} \implies Y_i = -X_i^{-(b-1)} \frac{\Omega(X_i^{-1})}{\Lambda'(X_i^{-1})}$$

# Forney algorithm during Chien search

Error magnitudes $Y_l$ can be computed by a Chien-search-like circuit:




At each step $i$, the values of $\Omega(\alpha^i)$ and $\Lambda'(\alpha^i)$ are available.

$Y_l$ can be computed by one division followed by multiplication by $\alpha^{-i(b-1)}$.