University of Arizona
ECE/OPTI 500C:
Photonic Communications Engineering I C
Fall 2010

Forward Error Correction (FEC)

By Ivan B. Djordjevic

THE UNIVERSITY OF ARIZONA
ece
electrical and computer engineering

Coding for Optical Channels

Important classes of codes:

- Linear block codes
- Cyclic codes
- Convolutional codes
- Turbo codes
- Low-density parity check codes

Linear codes
- Block codes
  - Cyclic
  - Noncyclic
- Convolutional codes

Nonlinear codes
- Block codes
- Trellis codes

- In (n,k) block code the channel encoder accepts information in successive k-bit blocks, for each blocks it adds n-k redundant bits that are algebraically related to the k message bits; thereby producing an overall encoded block of n bits (n>k), known as a *code word*)
- In convolutional code the encoding operation may be considered as the discrete-time convolution of the input sequence with the impulse response of the encoder.

Linear Block Codes

- **Linearity property**: A code is said to be linear if any two code words in the code can be added in modulo-2 arithmetic to produce a third code word.
- *Example*: (n,1) repetition code. The repetition code has two code words $\boldsymbol{x}_0$=(00 … 0) and $\boldsymbol{x}_1$=(11 … 1). The linear combination of two code words is a code word:

$$\boldsymbol{x}_0 + \boldsymbol{x}_0 = \boldsymbol{x}_0$$
$$\boldsymbol{x}_0 + \boldsymbol{x}_1 = \boldsymbol{x}_1 + \boldsymbol{x}_0 = \boldsymbol{x}_1$$
$$\boldsymbol{x}_1 + \boldsymbol{x}_1 = \boldsymbol{x}_0$$

- The set of code words from a linear block code forms a group under the addition operation, because all-zero code word serves as the identity element, and the code word itself serves as the inverse element. This is the reason why the linear block codes are also called the group codes.
- The linear block code (n,k) can be observed as a k-dimensional subspace of the vector space of all n-tuples over the binary filed GF(2).
- All n-tuples over GF(2) form the vector space:
  - The sum of two n-tuples $\boldsymbol{a}=(a_1\ a_2\ \ldots\ a_n)$ and $\boldsymbol{b}=(b_1\ b_2\ \ldots\ b_n)$ is clearly an n-tuple $\boldsymbol{c}=\boldsymbol{a}+\boldsymbol{b}$ $= (a_1+b_1\ a_2+b_2\ \ldots\ a_n+b_n)=(b_1+a_1\ b_2+a_2\ \ldots\ b_n+a_n)=\boldsymbol{b}+\boldsymbol{a}$. The all-zero vector $\boldsymbol{0}$=(0 0 … 0) is the identity element, and n-tuple $\boldsymbol{a}$ itself is the inverse element $\boldsymbol{a}+\boldsymbol{a}=0$. Therefore, the n-tuples form the Abelian group with respect to the addition operation.

– The scalar multiplication is defined by: $\alpha\boldsymbol{a}=(\alpha\ a_1\ \alpha a_2\ \ldots\ \alpha a_n)$, $\alpha\in GF(2)$

– Distributive laws:

$\alpha(\boldsymbol{a}+\boldsymbol{b})=\alpha\boldsymbol{a}+\alpha\boldsymbol{b}$

$(\alpha+\beta)\boldsymbol{a}=\alpha\boldsymbol{a}+\beta\boldsymbol{a},\ \forall\ \alpha,\ \beta\in GF(2)$

– Associate law:

$(\alpha\cdot\beta)\boldsymbol{a}=\alpha\cdot(\beta\boldsymbol{a})$

Clearly, the set of all *n*-tuples is a vector space over GF(2).

- The set of all code words from an (*n*,*k*) linear block code forms an abelian group under the addition operation. It can be shown, in a fashion similar to that above, that all code words of an (n,k) linear block codes form the vector space of dimensionality *k*.
  There exists k basis vectors (code words) such that every code word is a linear combination of these code words.
- *Example*: (*n*,1) repetition code: *C*={(0 0 … 0), (1 1 … 1)}. Two code words in *C* can be represented as linear combination of all-ones basis vector: (11 … 1)=1·(11 … 1), (00 … 0)= 1·(11 … 1)+1·(11 … 1)
- *Example*:
  (6,3) code: *C*={(000000), (001011),(010101),(011110),(100111),(101100),(110010),(111001)}. It can be easily shown that *C* is a 3-dimensional space with basis vectors $\boldsymbol{g}_0$=(100111), $\boldsymbol{g}_1$=(010101), g_2=(001011).

Generator Matrix for Linear Block Code

- Any code word $\boldsymbol{x}$ from the (n,k) linear block code can be represented as a linear combination of k basis vectors $\boldsymbol{g}_i$, i=0,1,..,k-1:

$$\boldsymbol{x} = m_0\boldsymbol{g}_0 + m_1\boldsymbol{g}_1 + \ldots + m_{k-1}\boldsymbol{g}_{k-1} = \boldsymbol{m}\begin{bmatrix}\boldsymbol{g}_0\\ \boldsymbol{g}_1\\ \ldots\\ \boldsymbol{g}_{k-1}\end{bmatrix}, \quad \boldsymbol{m} = \begin{pmatrix} m_0 & m_1 & \ldots & m_{k-1}\end{pmatrix}$$

k-bit message word

$$\boldsymbol{x} = \boldsymbol{m}\boldsymbol{G}, \quad \boldsymbol{G} = \begin{bmatrix}\boldsymbol{g}_0\\ \boldsymbol{g}_1\\ \ldots\\ \boldsymbol{g}_{k-1}\end{bmatrix}$$

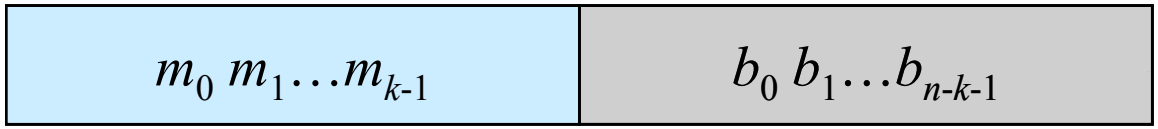
Generator matrix

- *Example*: Generator matrices for repetition (n,1) code, and (n,n-1) single-parity-check code:

$$\boldsymbol{G}_{\text{rep}} = \begin{bmatrix}11\ldots1\end{bmatrix}$$

$$\boldsymbol{G}_{\text{par}} = \begin{bmatrix}100\ldots01\\ 010\ldots01\\ \ldots\\ 000\ldots11\end{bmatrix}$$

Structure of systematic code word



Code word bits:

$$x_i = \begin{cases} m_i, & i = 0,1,\ldots,k-1 \\ b_{i-k}, & i = k,k+1,\ldots,n-1 \end{cases}$$

Parity bits:

$$b_i = p_{0i}m_0 + p_{1i}m_1 + \ldots + p_{k-1,i}m_{k-1}$$

$$p_{ij} = \begin{cases} 1, & \text{if } b_i \text{ depends on } u_j \\ 0, & \text{otherwise} \end{cases}$$

Matrix notation:

$$\boldsymbol{m} = \left[m_0 m_1 \ldots m_{k-1}\right] \quad \boldsymbol{b} = \left[b_0 b_1 \ldots b_{n-k-1}\right] \quad \boldsymbol{x} = \left[x_0 x_1 \ldots x_{n-1}\right]$$

Coefficient matrix:

$$\boldsymbol{P} = \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0,n-k-1} \\ p_{10} & p_{11} & \cdots & p_{1,n-k-1} \\ \cdots & & \cdots & \cdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} \end{bmatrix}$$

$$\Rightarrow \boldsymbol{x} = \left[\boldsymbol{m} \mid \boldsymbol{b}\right] = \boldsymbol{m}\left[\boldsymbol{I}_k \mid \boldsymbol{P}\right]$$

$\boldsymbol{I}_k$ – identity matrix

$\boldsymbol{G} = \left[\boldsymbol{I}_k \mid \boldsymbol{P}\right]$ – generator matrix

$$\Rightarrow \boldsymbol{x} = \boldsymbol{m}\boldsymbol{G}$$

Parity-Check Matrix for Linear Block Code

- Another useful matrix associated with the linear block codes is the parity-check matrix.
- Let us expand the matrix equation:

$$\boldsymbol{x} = \boldsymbol{m}\boldsymbol{G}$$

$$\Updownarrow$$

$$x_0 = m_0$$

$$x_1 = m_{k-1}$$

$$\ldots$$

$$x_{k-1} = m_{k-1}$$

$$x_k = m_0 p_{00} + m_1 p_{10} + \ldots + m_{k-1} p_{k-1,0}$$

$$x_{k+1} = m_0 p_{01} + m_1 p_{11} + \ldots + m_{k-1} p_{k-1,1}$$

$$\ldots$$

$$x_{n-1} = m_0 p_{0,n-k-1} + m_1 p_{1,n-k-1} + \ldots + m_{k-1} p_{k-1,n-k-1}$$

- The last *n-k* equations can be rewritten as:

$$x_0 p_{00} + x_1 p_{10} + \ldots + x_{k-1} p_{k-1,0} + x_k = 0$$
$$x_0 p_{01} + x_1 p_{11} + \ldots + x_{k-1} p_{k-1,0} + x_{k+1} = 0$$
$$\ldots$$
$$x_0 p_{0,n-k+1} + x_1 p_{1,n-k-1} + \ldots + x_{k-1} p_{k-1,n-k+1} + x_{n-1} = 0$$
$$\Updownarrow$$
$$\begin{bmatrix} x_0 & x_1 & \ldots & x_{n-1} \end{bmatrix} \begin{bmatrix} p_{00} & p_{10} & \ldots & p_{k-1,0} & 1 & 0 & \ldots & 0 \\ p_{01} & p_{11} & \ldots & p_{k-1,1} & 0 & 1 & \ldots & 0 \\ \ldots & & \ldots & & \ldots & & & \ldots \\ p_{0,n-k-1} & p_{1,n-k-1} & \ldots & p_{k-1,n-k-1} & 0 & 0 & \ldots & 1 \end{bmatrix}^T = \mathbf{0}$$
$$\Updownarrow$$
$$\boldsymbol{x}\boldsymbol{H}^T = \mathbf{0}, \quad \boldsymbol{H} = \begin{bmatrix} \boldsymbol{P}^T & \boldsymbol{I}_{n-k} \end{bmatrix}_{(n-k)xn}$$
$$\Rightarrow \boldsymbol{G}\boldsymbol{H}^T = \mathbf{0}$$
$$\boldsymbol{G}\boldsymbol{H}^T = \begin{bmatrix} \boldsymbol{I}_k & \boldsymbol{P} \end{bmatrix} \begin{bmatrix} \boldsymbol{P} \\ \boldsymbol{I}_{n-k} \end{bmatrix} = \boldsymbol{P} + \boldsymbol{P} = \mathbf{0}$$

- *Example*: Parity-Check Matrices for (*n*,1) repetition code and (*n*,*n*-1) single-parity check code

$$\boldsymbol{H}_{\text{rep}} = \begin{bmatrix} 100\ldots01 \\ 010\ldots01 \\ \ldots \\ 000\ldots11 \end{bmatrix} \qquad \boldsymbol{H}_{\text{par}} = \begin{bmatrix} 11\ldots1 \end{bmatrix}$$

Dual Code

- Every (*n,k*) linear block code with generator matrix ***G*** and parity-check matrix ***H*** has a dual code with generator matrix ***H*** and parity check matrix ***G***.
- *Example*: (*n*,1) repetition code and (*n*,*n*-1) single-parity check code are dual.

Syndrome: Definition and Properties

Received vector:

$$\boldsymbol{r} = \boldsymbol{x} + \boldsymbol{e}$$

$\boldsymbol{e}$ – error vector (pattern)

$$e_i = \begin{cases} 1 \text{ if an error occured in the } i\textit{th} \text{ location} \\ 0 \text{ otherwise} \end{cases}$$

Syndrome:

$$\boldsymbol{s} = \boldsymbol{r}\boldsymbol{H}^T$$

Properties:

1. ***The syndrome depends only on the error pattern, and not on the transmitted code word.***

   $\boldsymbol{s}=(\boldsymbol{x}+\boldsymbol{e})\boldsymbol{H}^T=\boldsymbol{x}\boldsymbol{H}^T+\boldsymbol{e}\boldsymbol{H}^T=\boldsymbol{e}\boldsymbol{H}^T$

2. ***All error patterns that differ by a code word have the same syndrome.***

For k message bits there are 2^k distinct vectors denoted by $\boldsymbol{x}_i$ (i=0,1,…,2^{k-1}). $\Rightarrow$ For any error pattern we can defined 2^k distinct vectors

$$\boldsymbol{e}_i=\boldsymbol{e}+\boldsymbol{x}_i,\ i=0,1,\ldots,2^k-1$$

- The *coset* of the code:

$\{\boldsymbol{e}_i,\ i=0,1,\ldots,2^k-1\}$

Since

$$\boldsymbol{e}_i\boldsymbol{H}^T=\boldsymbol{e}\boldsymbol{H}^T+\boldsymbol{x}_i\boldsymbol{H}^T=\boldsymbol{e}\boldsymbol{H}^T$$

Each coset of the code, having 2^k elements that differ at most by a code vector, is characterized by a unique syndrome.

– (n-k) elements of the syndrome are linear combinations of the n elements of the error pattern:

$$s_0=e_0+e_{n-k}p_{00}+e_{n-k-1}p_{10}+\ldots+e_{n-1}p_{k-1,0}$$

$$s_1=e_1+e_{n-k}p_{01}+e_{n-k-1}p_{11}+\ldots+e_{n-1}p_{k-1,1}$$

…

$$s_{n-k-1}=e_{n-k-1}+e_{n-k}p_{0,n-k-1}+e_{n-k-1}p_{1,n-k-1}+\ldots+e_{n-1}p_{k-1,n-k-1}$$

– The system of equations is underdetermined (more unknowns than equations)

– The knowledge of the syndrome reduces the search for the error pattern from 2^n to 2^{n-k} possibilities.

3. **The syndrome is the sum of those columns of the parity check matrix corresponding to the error locations.**

$$\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^T$$

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}_1 & \boldsymbol{h}_2 & \ldots & \boldsymbol{h}_n \end{bmatrix} \Rightarrow \boldsymbol{s} = \begin{bmatrix} e_1 & e_2 & \ldots & e_n \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_1^T \\ \boldsymbol{h}_2^T \\ \ldots \\ \boldsymbol{h}_n^T \end{bmatrix} = \sum_{i=1}^{n} e_i \boldsymbol{h}_i^T$$

4. **With syndrome decoding, an (*n*,*k*) linear block code can correct up to *t* errors per code word, providing that the Hamming bound is satisfied:**

$$2^{n-k} \geq \sum_{i=0}^{t} \binom{n}{i}$$

The total number of syndromes, including the all-zero syndrome, is 2^{n-k}, and each syndrome corresponds to a specific error pattern. For an *n*-bit code word there are *n* chooses *i* multiple-error patterns, where *i* is the number of error locations in error pattern **e**. The total number of all possible correctable error patterns:

$$\sum_{i=0}^{t} \binom{n}{i}$$

The total number of syndromes cannot be less than total number of possible error patterns.

• The binary code for which Hamming bound is satisfied with equality sign is called the **perfect code**.

Minimum Distance Considerations

•*Hamming distance* between two code words $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, $d(\boldsymbol{x}_1,\boldsymbol{x}_2)$, is defined as the number of locations in which their respective elements differ.

•*Hamming weight*, $w(\boldsymbol{x})$, of a code vector $\boldsymbol{x}$ is defined as the number of nonzero elements in the vectors.

•The *minimum distance*, d_{min}, of a linear block code is defined as the smallest Hamming distance between any pair of code vectors in the code.

•The *minimum distance of a linear block code is the smallest Hamming weight of the nonzero code vectors in the code*.

$\boldsymbol{xH}^T=\boldsymbol{0}$

$\boldsymbol{H}=[\boldsymbol{h}_1\ \boldsymbol{h}_2\ \ldots\ \boldsymbol{h}_n]$, $\boldsymbol{h}_i$ –*i*th column in $\boldsymbol{H}$-matrix

•The *minimum distance of a linear block code is defined by the minimum number of columns of the* $\boldsymbol{H}$*-matrix* whose sum is equal to the zero vector.

(a) **(b)**

(*a*) Hamming distance $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 2t + 1$. (*b*) Hamming distance $d(\boldsymbol{x}_i, \boldsymbol{x}_j) < 2t$. The received vector is denoted by **r**.

- An (*n*,*k*) *linear block code has the power to correct all error patterns of weight t or less, if, and only if,*

$d(\boldsymbol{x}_i,\boldsymbol{x}_j) \geq 2t+1$ for all $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

- An (*n*,*k*) *linear block code of minimum distance* d_{min} *can correct up to t errors if, and only if,*

$t \leq \lfloor 1/2(d_{min}-1) \rfloor$ (*)

where $\lfloor\ \rfloor$ denotes the largest integer less than or equal to the enclosed quantity.

Syndrome Decoding and Standard Array

- The 2^k code words partition the space of all received words into 2^k disjoint subsets. Any received word within subset is decoded as the unique code word. A standard array is a technique by which this partition can be achieved.
- **Standard Array Construction:**

1. Step 1. Write down 2^k code words as elements of the first row, with the all-zero codeword as the leading element.
2. Step 2. Repeat the steps 2(a) and 2(b) until all 2^n words are exhausted.
   (a) Out of the remaining unused *n*-tuples, select one with the least weight for the leading element of the next row (the *current row*).
   (b) Complete the current row by adding the leading element to each nonzero code word appearing in the first row and writing down the resulting sum in the corresponding column.

Standard array for an (*n*, *k*) block code:

	$x_1 = 0$	x_2	x_3	...	x_i	...	x_{2^k}
	e_2	$x_2 + e_2$	$x_3 + e_2$	...	$x_i + e_2$	...	$x_{2^k} + e_2$
Coset leaders	e_3	$x_2 + e_3$	$x_3 + e_3$	...	$x_i + e_2$	...	$x_{2^k} + e_3$
	...	...		...		...	...
	e_j	$x_2 + e_j$	$x_3 + e_j$	...	$x_i + e_j$	...	$x_{2^k} + e_j$
	...	...		...			...
	$e_{2^{n-k}}$	$x_2 + e_{2^{n-k}}$	$x_3 + e_{2^{n-k}}$	...	$x_i + e_{2^{n-k}}$	...	$x_{2^k} + e_{2^{n-k}}$

← **Code words**

- *Example*: Standard array of (6,3) code:

  C={(000000), (001011),(010101),(011110),(100111),(101100),(110010),(111001)}.

000000	**001011**	**010101**	**011110**	**100111**	**101100**	**110010**	**111001**
000001	001010	010100	011111	100110	101101	110011	111000
000010	001001	010111	011100	100101	101110	110000	111011
000100	001111	010001	011010	100011	101000	110110	111101
001000	000011	011101	010110	101111	100100	111010	110001
010000	011011	000101	001110	110111	111100	100010	101001
100000	101011	110101	111110	000111	001100	010010	011001
100001	101010	110100	111111	000110	001101	010011	011000

Standard array properties:

1. All n-tuples of row are distinct.
2. Each n-tuple appears exactly once in the standard array.
3. There are exactly 2^{n-k} rows in the standard array.
4. For perfect codes (satisfying the Hamming bound with equality sign) all n-tuples of weight t=int[$(d_{min}$-1)/2] or less appear as coset leaders (int[x] is the integer part of x).
5. For quasi-perfect codes, in addition to all n-tuples of weight t or less, some but not all n-tuples of weight t+1 appear as cosset leaders.
6. All elements in the same row (coset) have the same syndrome.
7. Elements in different rows have different syndromes.
8. There are 2^{n-k} different syndromes corresponding to 2^{n-k} rows.

- The key to the decoding using a standard array is to view the cosset leaders as error patterns caused by BSC. Let $\mathbf{e}_i$ represent the coset leader of ith row, any element in that row can be represented as $\mathbf{y}=\mathbf{x}+\mathbf{e}_i$, where $\mathbf{x}$ is the code word lying at the top of column in which $\mathbf{y}$ lies, and it is the closest code word to $\mathbf{y}$:

$$d_H(\mathbf{y},\mathbf{x}) = w(\mathbf{y}+\mathbf{x}) = w(\mathbf{e}_i + \mathbf{x} + \mathbf{x}) = w(\mathbf{e}_i)$$

$$d_H(\mathbf{y},\mathbf{x}_1) = w(\mathbf{y}+\mathbf{x}_1) = w(\mathbf{e}_i + \mathbf{x} + \mathbf{x}_1) = w(\mathbf{e}_i + \mathbf{x}_2)$$

$$w(\mathbf{e}_i) \le w(\mathbf{e}_i + \mathbf{x}_2) \Rightarrow d_H(\mathbf{y},\mathbf{x}) = w(\mathbf{e}_i) \le d_H(\mathbf{y},\mathbf{x}_1) = w(\mathbf{e}_i + \mathbf{x}_2)$$

- **The probability of word error**:

$$P_w(e) = 1 - \sum_{i=0}^{n} w_i \, p^i (1-p)^{n-i}$$

Crossover probability of BSC

The number of coset leaders of weight i

- The weight distribution of the coset leaders: w_i, i=0,1,…,n
- *Example*: The weight distribution of coset leaders in (6,3) code are w_0=1, w_1=6, w_2=1, w_i=0, i=3,…,6; leading to the word error probability:

$$P_w(e) = 1 - (1-p)^6 - 6p(1-p)^5 - p^2(1-p)^4$$

$$P_w(e) \approx 14p^2(1-p)^4\Big|_{p=10^{-3}} = 1.4 \cdot 10^{-5}$$

Because all single errors and one double error are correctable, the remaining 14 double errors dominate for small p

Decoding procedure (**syndrome decoding**):

- For the received vector $\boldsymbol{y}$, compute the syndrome $\boldsymbol{s}=\boldsymbol{y}\boldsymbol{H}^T$. Notice that one-to-one correspondence can be established between the syndromes and error patterns (Property 7), leading to the lookup table containing the error pattern.
- Within the coset characterized by the syndrom $\boldsymbol{s}$, identify the coset leader (i.e., the error pattern with the largest probability of occurence); $\boldsymbol{e}_0$.
- Compute the code vector

  $\boldsymbol{x}=\boldsymbol{y}+\boldsymbol{e}_0$ as the decoding version of the received vector $\boldsymbol{y}$.

- *Example*: $\boldsymbol{y}=(110100) \Rightarrow \boldsymbol{s}=\boldsymbol{y}\boldsymbol{H}^T=(111) \Rightarrow \boldsymbol{e}_0=(100000) \Rightarrow \boldsymbol{x}=\boldsymbol{y}+\boldsymbol{e}_0=(011110)$

Syndrome (the address to the lookup table)	Error Pattern
000	000000
001	000001
010	000010
011	000100
100	001000
101	010000
110	100001
111	100000

The lookup table for (6,3) code

Hamming Codes

- A family of (n,k) linear block codes with following parameters:
  - Block length: $n=2^m-1$
  - Number of message bits: $k=2^m-m-1$
  - Number of parity bits: $n-k=m$

  where $m\geq 3$ are known as Hamming codes.

Example: **(7,4) Hamming code**

$$\boldsymbol{G}=\left[\begin{array}{c|c} 110 & 1000 \\ 011 & 0100 \\ 111 & 0010 \\ 101 & 0001 \end{array}\right] \Rightarrow \boldsymbol{H}=\left[\begin{array}{c|c} 100 & 1011 \\ 010 & 1110 \\ 001 & 0111 \end{array}\right] \Rightarrow d_{\min}=3 \quad \text{since} \quad \begin{bmatrix}0\\1\\0\end{bmatrix}+\begin{bmatrix}0\\0\\1\end{bmatrix}+\begin{bmatrix}0\\1\\1\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix}$$

- ***Hamming codes are single-error correcting binary perfect codes.***

(7,4) Hamming *code words*:

Message Word	*Code Word*	*Weight of Code Word*
0000	0000000	0
0001	1010001	3
0010	1110010	4
0011	0100011	3
0100	0110100	3
0101	1100101	4
0110	1000110	3
0111	0010111	4
1000	1101000	3
1001	0111001	4
1010	0011010	3
1011	1001011	4
1100	1011100	4
1101	0001101	3
1110	0101110	4
1111	1111111	7

Decoding table for (7,4) Hamming code:

Syndrome	*Error Pattern*
000	0000000
100	1000000
010	0100000
001	0010000
110	0001000
011	0000100
111	0000010
101	0000001

-Code word [11100010] is sent, and the received vector [1100010] is with error in the third bit.

$$\boldsymbol{s} = [1100010 \quad] \begin{bmatrix} 100 \\ 010 \\ 001 \\ 110 \\ 011 \\ 111 \\ 101 \end{bmatrix} = [001 \quad] \Rightarrow \boldsymbol{e} = [0010000 \quad]$$

Coding Gain

- *Coding gain* refers to the savings attainable in the energy per information bit to noise spectral density ratio (E_b/N_0) required to achieve a given bit error probability when coding is used compared to that with no coding.

$$nE_c = kE_b \Rightarrow E_c = \frac{k}{n}E_b = rE_b$$

$$BPSK:$$

$$p = Q\left(\sqrt{\frac{2E_c}{N_0}}\right) = Q\left(\sqrt{\frac{2rE_b}{N_0}}\right)$$

$$\text{At high signal-to-noise ratio:}$$

$$P_w(e) \approx \binom{N}{t+1} p^{t+1}(1-p)^{N-t+1}$$

$$\approx \binom{N}{t+1} p^{t+1}$$

$$\text{Bit error probability:}$$

$$P_b \approx \frac{2t+1}{n} P_w(e)$$

$$\Rightarrow P_b \approx c(n,t)p^{t+1}$$

$$\text{Since } Q(x) \le \frac{1}{2}\exp\left(\frac{-x^2}{2}\right)$$

$$\Rightarrow P_b \approx \frac{c}{2}\left[\exp\left(\frac{-rE_b}{N_0}\right)\right]^{t+1}$$

Uncoded bit error probability:

$$P_{b,un} \approx \frac{1}{2}\exp\left(-\frac{E_b}{N_0}\right)$$

Coding gain:

$$G_h = \frac{\left(E_b/N_0\right)_{un}}{\left(E_b/N_0\right)_c} \approx r(t+1)$$

Hard-decision:

$$G_h[\text{dB}] = 10\log_{10}[r(t+1)]$$

Soft-decision:

$$G_s[\text{dB}] = 10\log_{10}[rd_{\min}]$$

Example: **Coding Gain for (7,4) Hamming Code**

$$P_w(e) = 1-(1-p)^7 - 7p(1-p)^6$$
$$= \sum_{i=2}^{7}\binom{7}{i}p^i(1-p)^{7-i}$$
$$\approx 21p^2$$

$$P_b \approx \frac{3}{7}P_w(e)$$

α – given a certain BER

$$\Rightarrow P_w(e) = \frac{7}{3}\alpha \approx 21p^2$$

$$\Rightarrow p = \sqrt{\alpha}/3 = Q\left(\sqrt{\frac{2rE_b}{N_0}}\right)$$

$\Rightarrow$ Required E_b/N_0 to achieve the given BER

Cyclic Codes

1. *Linearity property*: The sum of any two code words in the code is also a code word.
2. *Cyclic property*: Any cyclic shift of a code word in the code is also a code word.

$$(c_0 c_1 \ldots c_{n-1})$$
$$(c_{n-1} c_0 \ldots c_{n-2})$$
$$(c_{n-2} c_{n-1} c_0 \ldots c_{n-3})$$
$$\ldots$$
$$(c_1 c_2 \ldots c_{n-1} c_0)$$

Code word polynomial:

$$(c_0 c_1 \ldots c_{n-1}) \Leftrightarrow c(X) = c_0 + c_1 X + c_2 X^2 + \ldots + c_{n-1} X^{n-1}$$

$$c^{(i)}(X) = X^i c(X) \bmod (X^n + 1) - \text{also a code word polynomial}$$

Generator Polynomial

- The polynomial X^n+1 and its factors play a major role in the generation of cyclic codes.
- *Generator polynomial*

$$g(X) = 1 + \sum_{i=1}^{n-k-1} g_i X^i + X^{n-k}$$

- $g(X)$ is a factor of X^n+1, $\deg(g(X))=n-k$, $g(X)$ is the *least degree polynomial* in the code

•A cyclic code is uniquely determined by the generator polynomial:

$$c(X) = a(X)g(X), \quad c(X) - \text{a code word polynomial}$$

Systematic cyclic code

- message polynomial:

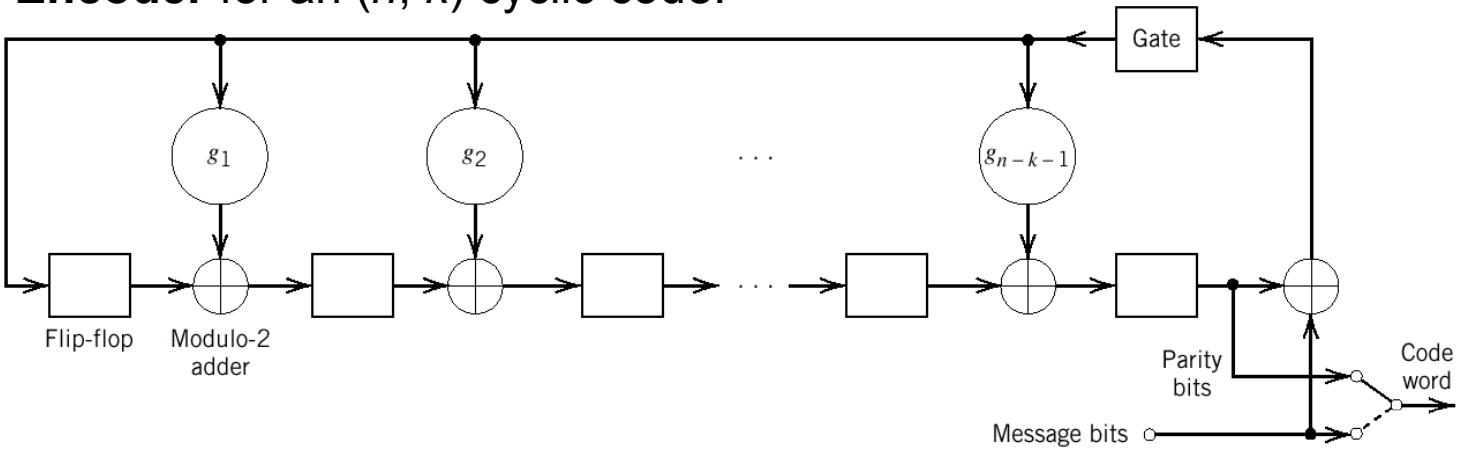
$$m(X) = m_0 + m_1 X + \ldots + m_{k-1} X^{k-1}$$

$$\frac{X^{n-k} m(X)}{g(X)} = a(X) + \frac{b(X)}{g(X)}, \quad b(X) - \text{the reminder}$$

$$b(X) = b_0 + b_1 X + \ldots + b_{n-k-1} X^{n-k-1}$$

$$\underline{C(X) = X^{n-k} m(X) + b(X)}$$

Encoder for an (*n*, *k*) cyclic code.



Syndrome Calculation

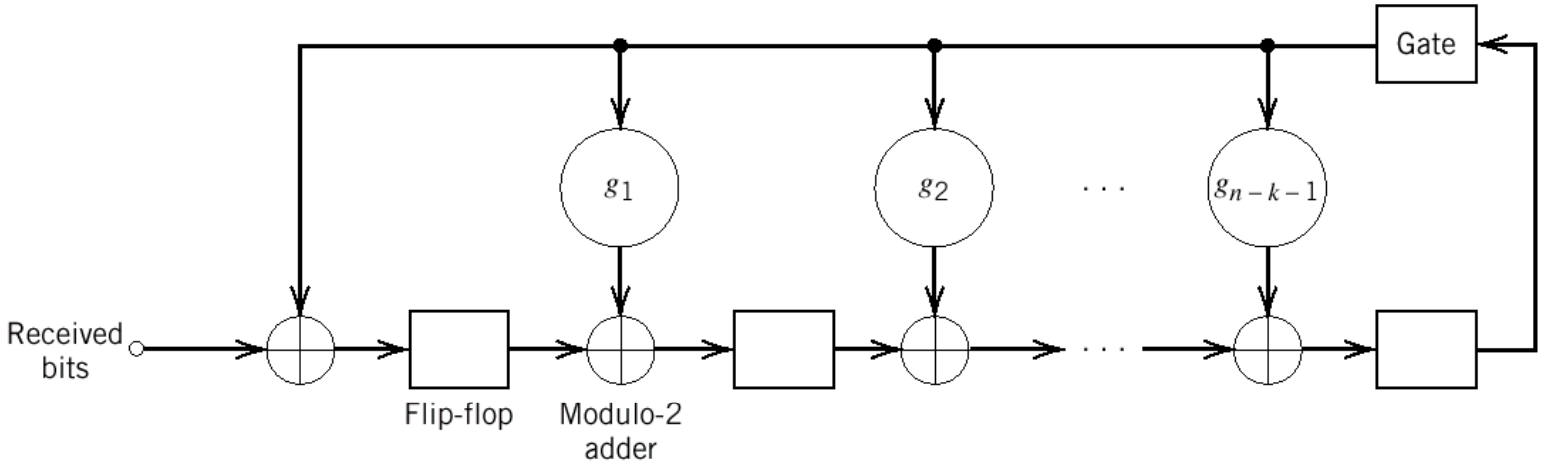
Received word polynomial:

$$r(X) = r_0 + r_1 X + \ldots + r_{n-1} X^{n-1}$$

$$\frac{r(X)}{g(X)} = q(X) + \frac{s(X)}{g(X)}, \quad s(X) - \text{the reminder}$$

$$\deg[s(X)] = n - k - 1$$

$$s(X) - \textit{the syndrome polynomial}$$



Syndrome calculator for (*n*, *k*) cyclic code.

- Properties of the syndrome polynomial:

1. The syndrome of received word polynomial is also the syndrome of corresponding error polynomial.
2. Let $s(X)$ be the syndrome of received word polynomial $r(X)$. Then, the syndrome of $Xr(X)$, a cyclic shift of $r(X)$, is $Xs(X)$.
3. The syndrome polynomial s(X) is identical to the error polynomial e(X), assuming that the errors are confined to the $(n-k)$ parity-check bits of the received word polynomial $r(X)$.

Parity-Check Polynomial

$$h(X) = 1 + \sum_{i=1}^{k-1} g_i X^i + X^k$$

- The generator polynomial $g(X)$ and the parity-check polynomial $h(X)$ are factors of the X^n+1:

$$g(X)h(X) \bmod \left(X^n + 1\right) = 0$$

Generator and Parity-Check Matrices

- n-tuples pertaining to the k polynomials $g(X)$, $Xg(X),\ldots,X^{k-1}g(X)$ may be used in rows of of the k-by-n generator matrix ***G***.
- n-tuples pertaining to the $(n-k)$ polynomials $X^k h(X^{-1})$, $X^{k+1}h(X^{-1}),\ldots,X^{n-1}h(X^{-1})$ may be used in rows of the $(n-k)$-by-n parity-check matrix ***H***.

Hamming Codes Revisited [(7,4) Cyclic Code Example]

$X^7+1=(1+X)(1+X^2+X^3)(1+X+X^3)$

$g(X) = 1 + X + X^3$

$h(X)=(1+X)(1+X^2+X^3)=1+X+X^2+X^4$

Message sequence: 1001

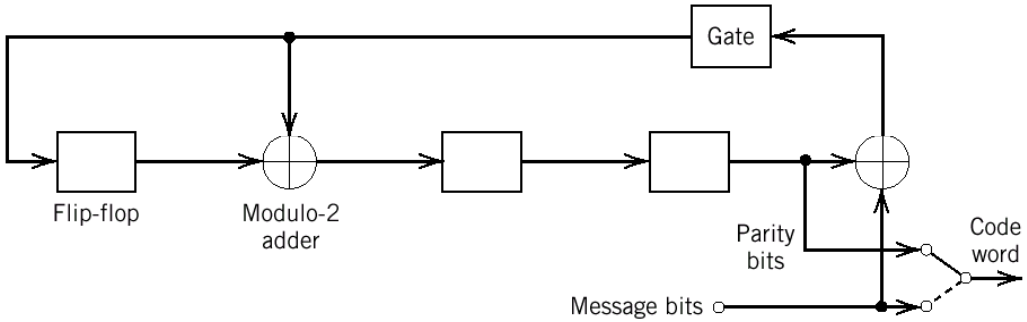
$m(X)=1+X^3$

$X^{n-k}\mathrm{m}(X)=X^3\mathrm{m}(X)=X^3+X^6$

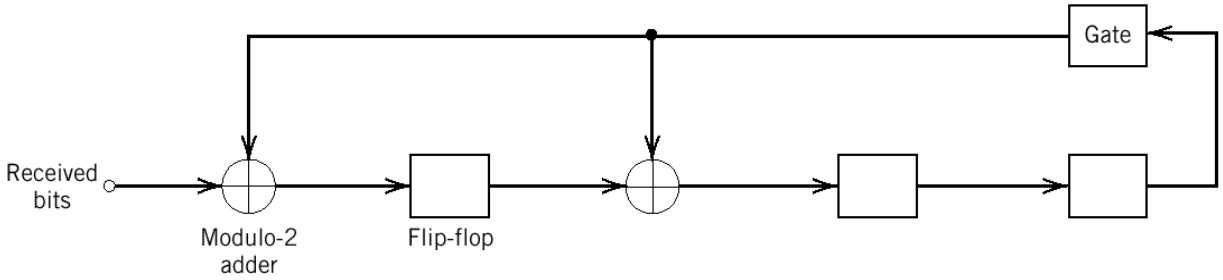
$$\frac{X^3+X^6}{1+X+X^3} = X + X^3 + \frac{X+X^2}{1+X+X^3}$$

$$\Rightarrow b(X) = X + X^2$$

$$C(X) = X^{n-k}m(X) + b(X) = X + X^2 + X^3 + X^6 \Leftrightarrow 0111001$$



Encoder for the (7, 4) cyclic code generated by $g(X) = 1 + X + X^3$.



Syndrome calculator for the (7, 4) cyclic code generated by the polynomial $g(X) = 1 + X + X^3$.

$$g(X) = 1 + X + X^3$$

$$Xg(X) = X + X^2 + X^4$$

$$X^2 g(X) = X^2 + X^3 + X^5$$

$$X^3 g(X) = X^3 + X^4 + X^6$$

$$\boldsymbol{G}' = \begin{bmatrix} 1101000 \\ 0110100 \\ 0011010 \\ 0001101 \end{bmatrix}$$

$$\Rightarrow \boldsymbol{G} = \begin{bmatrix} 1101000 \\ 0110100 \\ 1110010 \\ 1010001 \end{bmatrix}$$

$$X^4 h\left(X^{-1}\right) = 1 + X^2 + X^3 + X^4$$

$$X^5 h\left(X^{-1}\right) = X + X^3 + X^4 + X^5$$

$$X^6 h\left(X^{-1}\right) = X^2 + X^3 + X^5 + X^6$$

$$\boldsymbol{H}' = \begin{bmatrix} 1011100 \\ 0101110 \\ 0010111 \end{bmatrix}$$

$$\Rightarrow \boldsymbol{H} = \begin{bmatrix} 1001011 \\ 0101110 \\ 0010111 \end{bmatrix}$$

Example: **Maximal-Length Codes**

Maximal-length codes are dual of Hamming codes:

- Block length: $n=2^m-1$ ($m \geq 3$)
- Number of message bits: $k=m$
- Minimum distance: $d_{\min}=2^{m-1}$

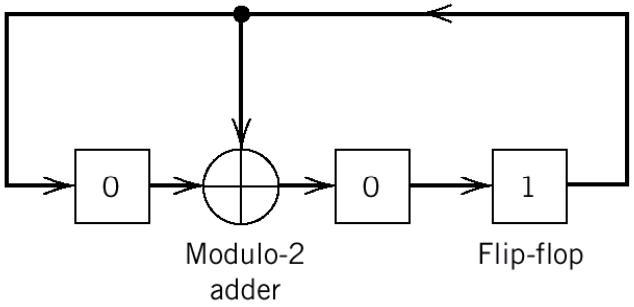
$$g(X) = \frac{1+X^n}{h(X)}$$

$$\deg[h(X)] = m$$

$$(7,3)\ \text{maximal-length code:}$$

$$h(X) = 1 + X + X^3$$

$$g(X) = 1 + X + X^2 + X^4$$



Encoder for the (7, 3) maximal-length code; the initial state of the encoder is shown in the figure.

Other Cyclic Codes

Cyclic Redundancy Check Codes (CRC Codes)

- Extremely well suited for error detection
- Error burst of length *B*: a contiguous sequence of *B* bits in which the first and and the last bits or any other intermediate bits are received in error.
- (*n*,*k*) CRC codes are capable of detecting:

1. All error bursts of length $n-k$.
2. A fraction of error bursts of length equal to $n-k+1$; the fraction equals $1-2^{-(n-k-1)}$.
3. A fraction of error of length greater than $n-k+1$; the fraction equals $1-2^{-(n-k-1)}$.
4. All combinations of $d_{min}-1$ (or fewer) errors.
5. All error patterns with an odd number of errors if the generator polynomial $g(X)$ for the code has an even number of nonzero coefficients.

CRC Codes	*Generator polynomial*	*n-k*
CRC-12 code	$1+X+X^2+X^3+X^{11}+X^{12}$	12
CRC-16 code (USA)	$1+X^2+X^{15}+X^{16}$	16
CRC-ITU	$1+X^5+X^{12}+X^{16}$	16

Bose-Chaudhuri-Hocquenghem (BCH) Codes

- *Primitive BCH codes (t-*error correcting codes*)*:
  - Block length: $n=2^m-1$ ($m\geq 3$)
  - Number of message bits: $k\geq n-mt$
  - Minimum distance: $d_{min}\geq 2t+1$

Binary BCH codes of length up to 2^5-1:

n	*k*	*t*	*Generator Polynomial*								
7	4	1								1	011
15	11	1								10	011
15	7	2							111	010	001
15	5	3						10	100	110	111
31	26	1								100	101
31	21	2						11	101	101	001
31	16	3				1	000	111	110	101	111
31	11	5			101	100	010	011	011	010	101
31	6	7	11	001	011	011	110	101	000	100	111

$$111010001 \Leftrightarrow g(X) = X^8 + X^7 + X^6 + X^4 + 1$$

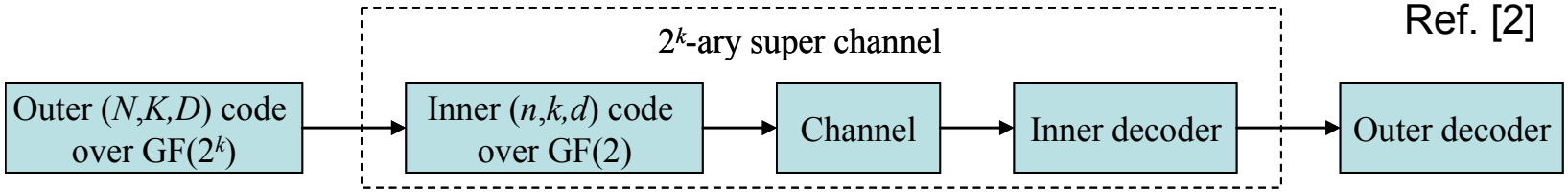
Reed-Solomon Codes (RS Codes)

- RS codes are an important subclass of *nonbinary* BCH codes
- A *t*-error correcting RS code parameters:
  - Block length: $n=2^m-1$ symbols
  - Message size: k symbols
  - Parity-check size: $n-k=2t$ symbols
  - Minimum distance: $d_{min}=2t+1$ symbols

Concatenated Codes

- The *concatenated code* (proposed by Forney), is an (*Nn*,*Kk*≥*Dd*) code with the minimum distance of at least *Dd*:

Ref. [2]



- For example, RS(255,239,8) code can be combined with the (12,8,3) single parity check code in the concatenation scheme (12·255,239·8,≥24).

Interleaved Codes

- Two RS codes can be combined in a concatenated scheme by ***interleaving***.
- An *interleaved code* is obtained by taking L codewords (of length N) of a given code $\boldsymbol{x}_j=(x_{j1},x_{j2},\ldots,x_{jN})$ (j=1,2,…,L), and forming the new codeword by interleaving the L codewords as follows $\boldsymbol{y}_i=(x_{11},x_{21},..,x_{L1},\ x_{12},x_{22},..,x_{L2},\ldots,x_{1N},x_{2N},..,x_{LN})$.
- The process of interleaving can be visualized as the process of forming an LxN matrix of L codewords written row by row and transmitting the matrix column by column, as given below

$$x_{11}\ x_{12}\ \ldots\ x_{1N}$$

$$x_{21}\ x_{22}\ \ldots\ x_{2N}$$

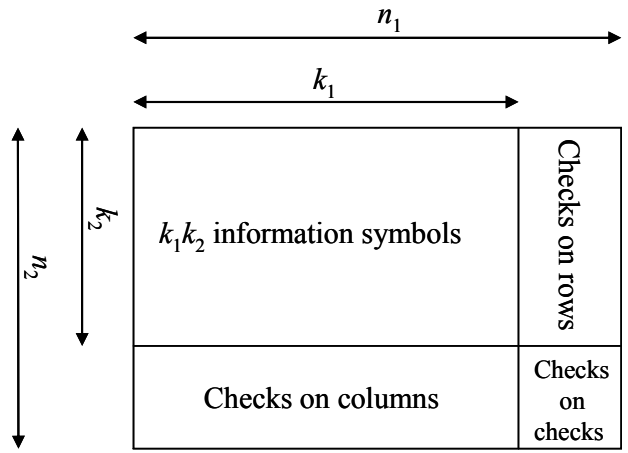
$$\ldots\ldots\ldots\ldots$$

$$x_{L1}\ x_{L2}\ \ldots\ x_{LN}$$

- The parameter L is known as the *interleaving degree*.

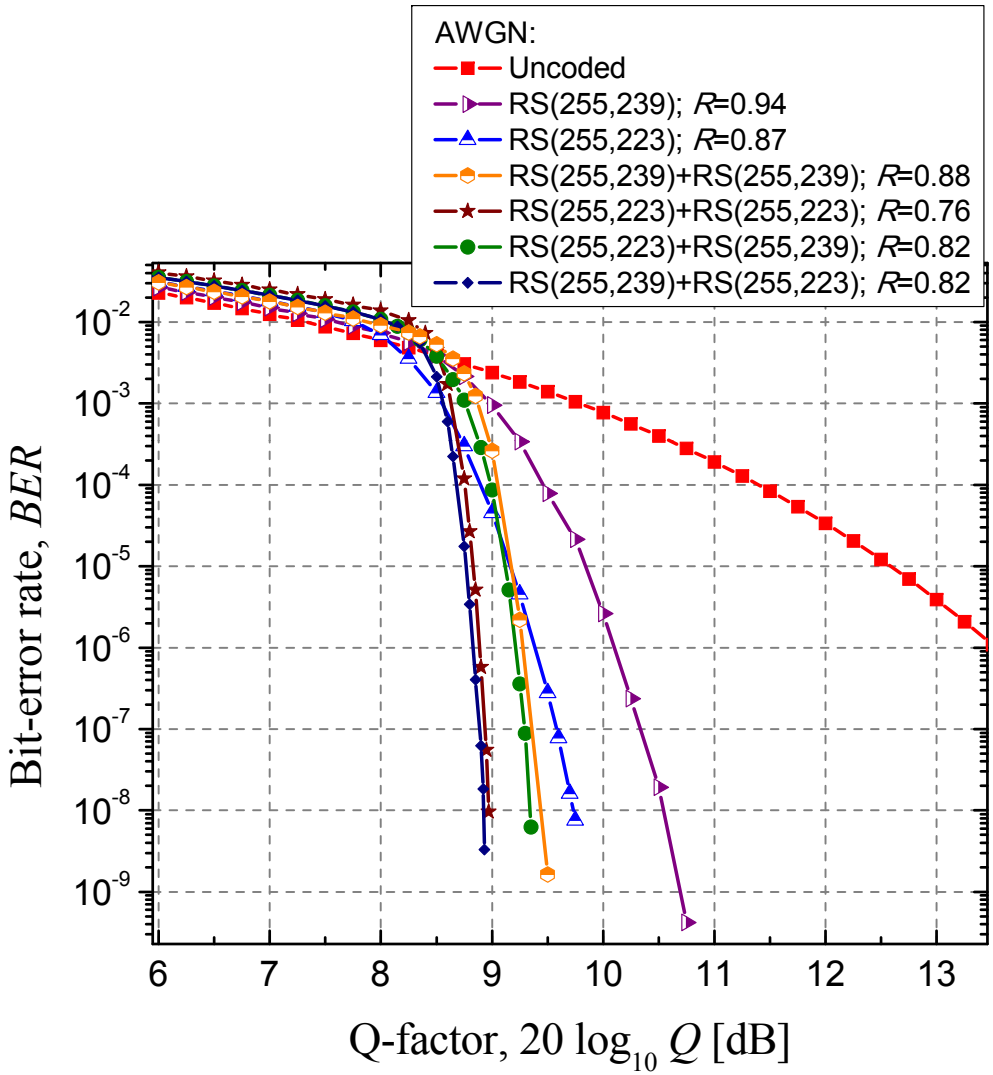
Product Codes

- Another way to deal with burst errors is to arrange two RS codes into a ***product code***:



- A product code (proposed by Elias) is an (n_1n_2,k_1k_2,d_1d_2) code in which codewords form an n_1xn_2 array such that each row is a codeword from an (n_1,k_1,d_1) code C_1, and each column is a codeword from an (n_2,k_2,d_2) code C_2; with n_i, k_i and d_i (i=1,2) being the codeword length, dimension and minimum distance, respectively, of ith component code.
- Both binary (such as binary BCH codes) and nonbinary codes (such as RS codes) may be arranged in the turbo product manner.
- It is possible to show that the minimum distance of a product codes is the product of minimum distances of component codes.
- It is straightforwardly to show that the product code is able to correct the burst error of length b=max(n_1b_2,n_2b_1), where b_i is the burst error capability of component code i=1,2.

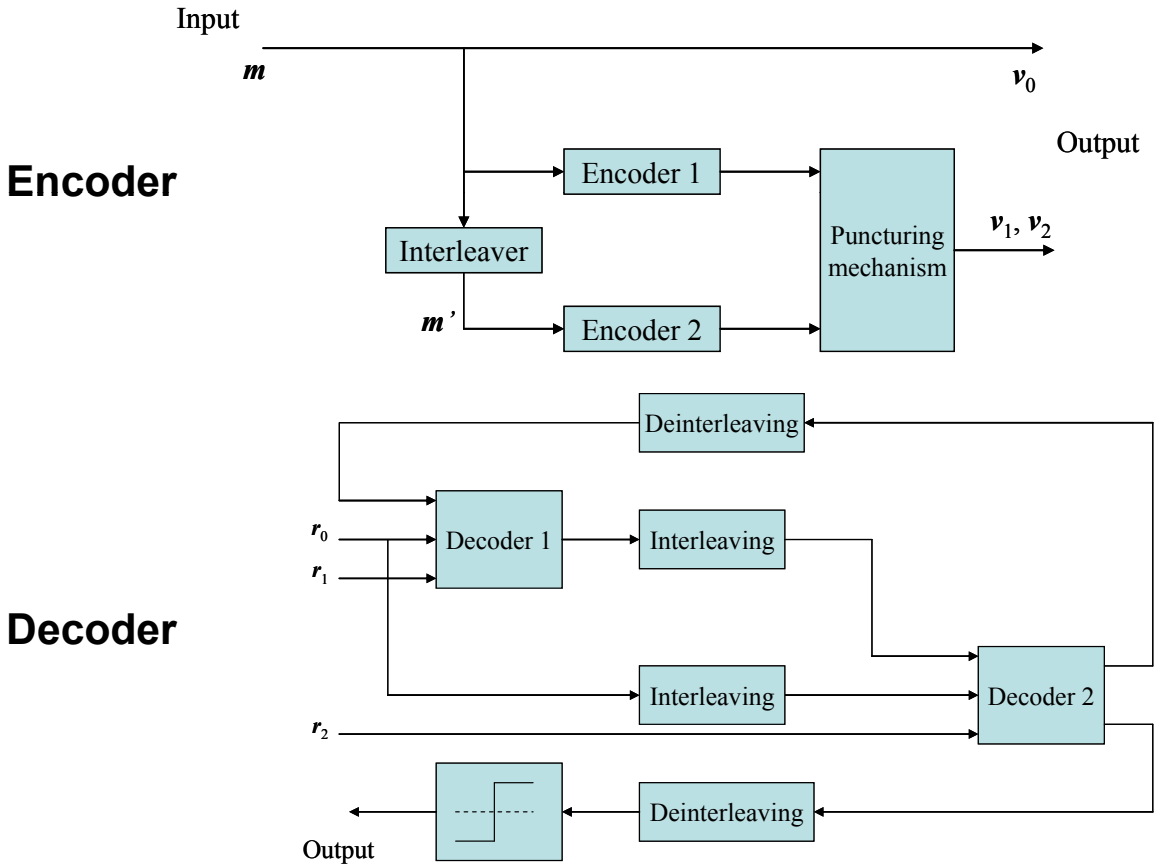
BER Performance



THE UNIVERSITY OF ARIZONA
ece
electrical and computer engineering

Turbo Codes

- The turbo codes can be considered as the generalization of the concatenation of codes in which, during iterative decoding, the decoders interchange the soft messages certain number of times.
- One possible implementation of turbo encoder and decoder based on systematic convolutional or block codes is given below.

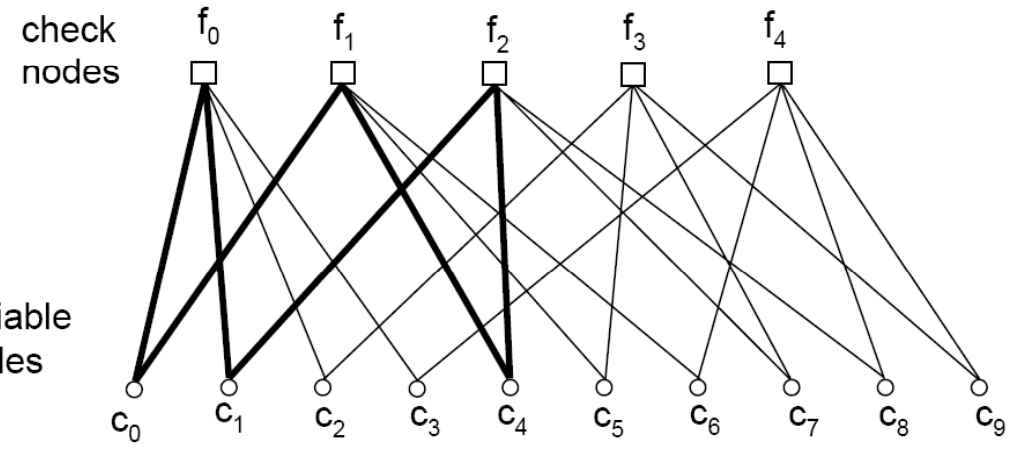


Low-Density Parity Check (LDPC) Codes

- Definition. A low-density parity-check (LDPC) code is a linear block code for which the parity-check matrix H has a low density of 1's.
- Definition. A regular (n, k) LDPC code is a linear block code whose parity-check matrix H contains exactly W_c 1's per column and exactly $W_r = W_c(n/m)$ 1's per row, where $W_c << m$.
- The code rate $r = k/n$ can be computed from $r = \frac{W_r - W_c}{W_r} = 1 - \frac{W_c}{W_r}$
- $W_c \geq 3$ is necessary for a good LDPC codes (Gallager)
- Definition. A bipartite graph (Tanner graph) is a graph (nodes or vertices connected by undirected edges) whose nodes may be separated into two classes, and where edges may only connect two nodes not residing in the same class.
  - Tanner graph of a code is drawn according to the following rule: check node c is connected to variable node v whenever element h_{cv} in H is a 1
- Girth: the shortest cycle in Tanner graph.

- **Example.** (10, 5) block code with $W_c = 2$ and $W_r = W_c(n/m) = 4$.

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

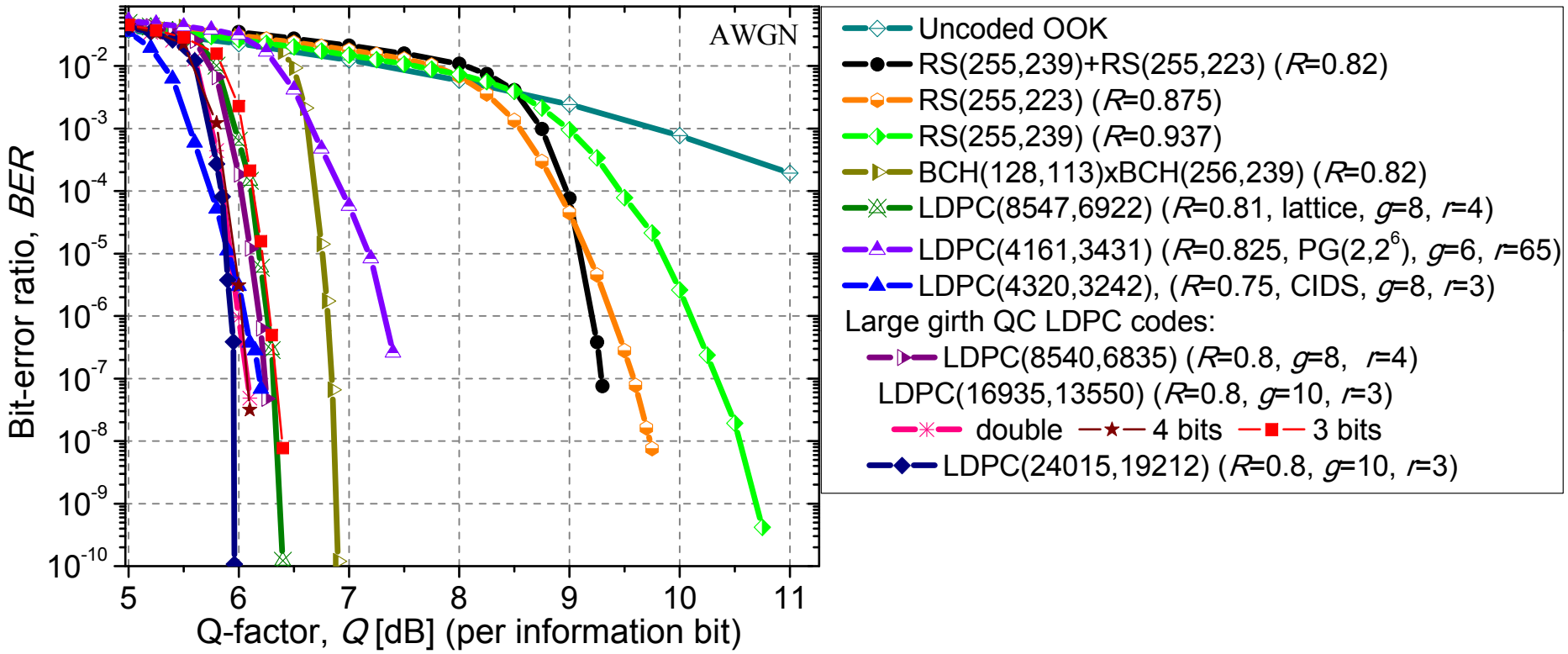


Quasi-Cyclic (QC)-LDPC Codes

- **The parity check-matrix of QC-LDPC codes can be represented by:**

$$H = \begin{bmatrix} I & I & I & \cdots & I \\ I & P^{S[1]} & P^{S[2]} & \cdots & P^{S[c-1]} \\ I & P^{2S[1]} & P^{2S[2]} & \cdots & P^{2S[c-1]} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ I & P^{(r-1)S[1]} & P^{(r-1)S[2]} & \cdots & P^{(r-1)S[c-1]} \end{bmatrix}$$

BER Performance of Large-Girth LDPC Codes



- The girth-10 LDPC(24015,19212) code of rate 0.8 (and column weight 3) outperforms the concatenation RS(255,239)+RS(255,223) (of rate 0.82) by 3.35 dB and RS(255,239) by 4.75 dB both at BER of 10^{-7}.

References

- S. Lin, D. Costello, *Error Control Coding: Fundamentals and Applications*, 2nd Ed., Prentice Hall, 2004.
- S. Haykin, *Communication Systems*, 4th Ed., John Wiley & Sons, Inc., 2001.
- J. B. Anderson, S. Mohan, *Source and Channel Coding: An Algorithmic Approach*, Kluwer Academic Publishers, 1991.
- I. B. Djordjevic, W. Ryan, and B. Vasic, *Coding for Optical Channels*. Springer, Mar. 2010