Chapter 10
Error-Control Coding

Problems: (pp.696-702)

10.4	10.7	10.12
10.15	10.17	10.25

Chapter 10
Error-Control Coding

第十章 纠错控制编码

Chapter 10
Error-Control Coding

- Topics: 差错控制编码、纠错码、信道编码
  - Error-control coding --- Channel coding
  - Important codes
    - Linear block codes (Cyclic codes)
    - Convolutional codes
    - Compound codes (turbo codes, low-density parity-check codes & irregular codes)

10.1 Introduction

- Task --- transmitting information

  at
  - a rate(velocity)
  - a level of reliability(quality)

- Why *error-control coding?*

  For a fixed E_b/N_0, it is the only practical option available for changing data quality from problematic to acceptable.

10.1 Introduction

- System parameters:
  - transmitted signal power
  - channel bandwidth

\+ power spectral density of receiver noise

Reduce transmitted power or hardware costs

Signal energy per bit/noise power spectral density (E_b/N_0)

(For fixed *BER*)

bit error rate

error control coding

(For fixed E_b/N_0)

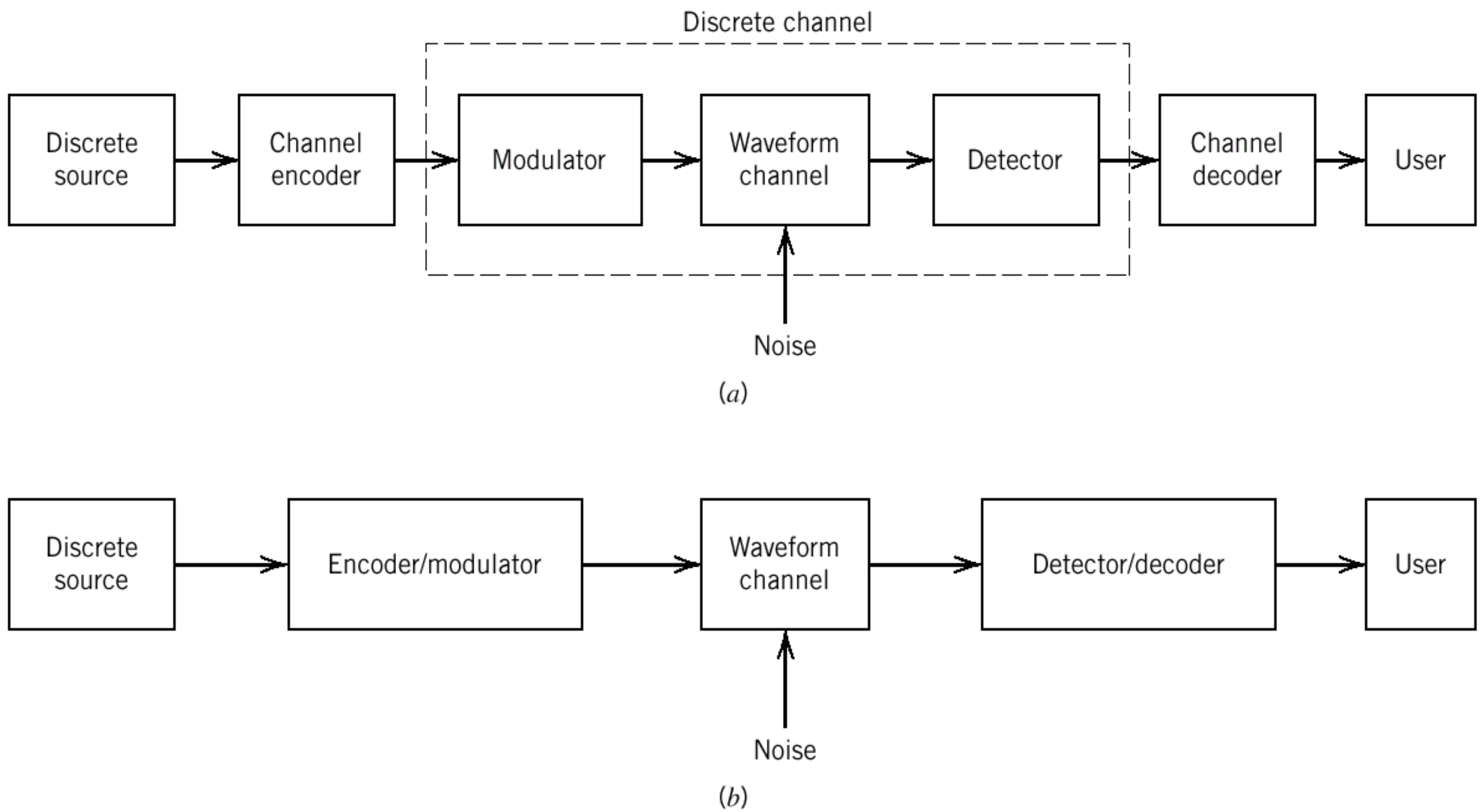


Figure 10.1

Simplified models of digital communication system.
(*a*) Coding and modulation performed separately.
(*b*) Coding and modulation combined.

10.1 Introduction

- In fig. 10.1a, channel coding and modulation are performed separately.

- In fig. 10.1b, channel coding and modulation are combined(TCM). ---> bandwidth efficiency is increased

10.1 Introduction

Channel encoder -- adds redundancy according to a prescribed rule

Channel decoder -- exploit the redundancy to decide actually transmitted message bits

⇩

Advantage -- minimize the effect of channel noise (reduce the BER)

Disadvantage -- increase transmission bandwidth and system complexity

10.1 Introduction

- Classification :
  - block codes -- absence of memory
    - (n, k) block code --- (n-k) redundant bits
    - code rate r=k/n <1
    - channel data rate $R_0=(n/k)*R_s$ (source data rate) >1
  - convolutional codes -- presence of memory
    - (n, k, N) convolutional code
    - the input sequence $\otimes$ the impulse response of the encoder (duration=memory of the encoder N)
    - $\otimes$ -- discrete time convolution

linear √
nonlinear

systematic
nonsystematic

10.1 Introduction

- FEC (Feed-forward error correction)
  - redundancy used for detection & correction of errors in receiver (channel coding)
  - one-way link & decoding complexity
  - wider in application
- ARQ (automatic-repeat request) 检错码
  - redundancy used only for detection of errors error --> repeat transmission
  - half-duplex or full-duplex links (feed-back channel)
  - computer communication systems

10.1 Introduction

- Types of ARQ
  - stop-and-wait strategy (half-duplex link)
  - continuous ARQ with pullback (duplex link)
  - continuous ARQ with selective repeat (duplex link)
- The above three types of ARQ offer trade-off between the need for a half-duplex or full-duplex link and data throughout.

10.2 Discrete-Memoryless Channels

- **Memoryless waveform channel:** the detector output in a given interval depends only one the signal transmitted in that interval (see fig. 10.1a)
- **Discrete memoryless channel:** the modulator + the waveform channel + the detector
- **Transition probabilities** $p(j/i)$ describe a discrete memoryless channel completely

10.2 Discrete-Memoryless Channels

- Binary symmetric channel

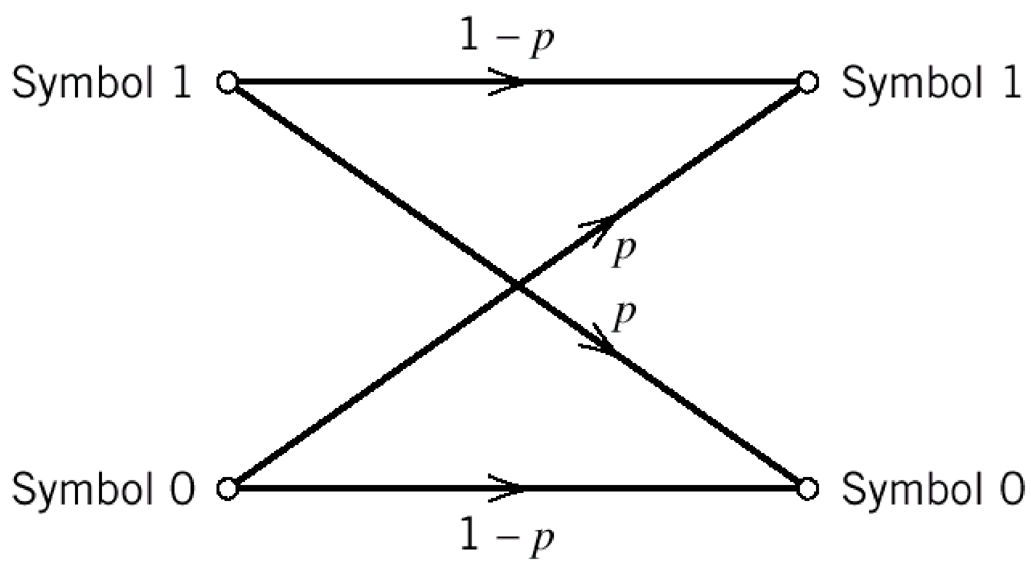


Figure 10.2 Transition probability diagram of binary symmetric channel.

10.2 Discrete-Memoryless Channels

- Hard decision decoder -- algebraic decoder
  - simplicity of implementation
  - irreversible loss of information

- Soft decision decoder -- probabilistic decoder
  - complicates the implementation
  - offers significant improvement in performance

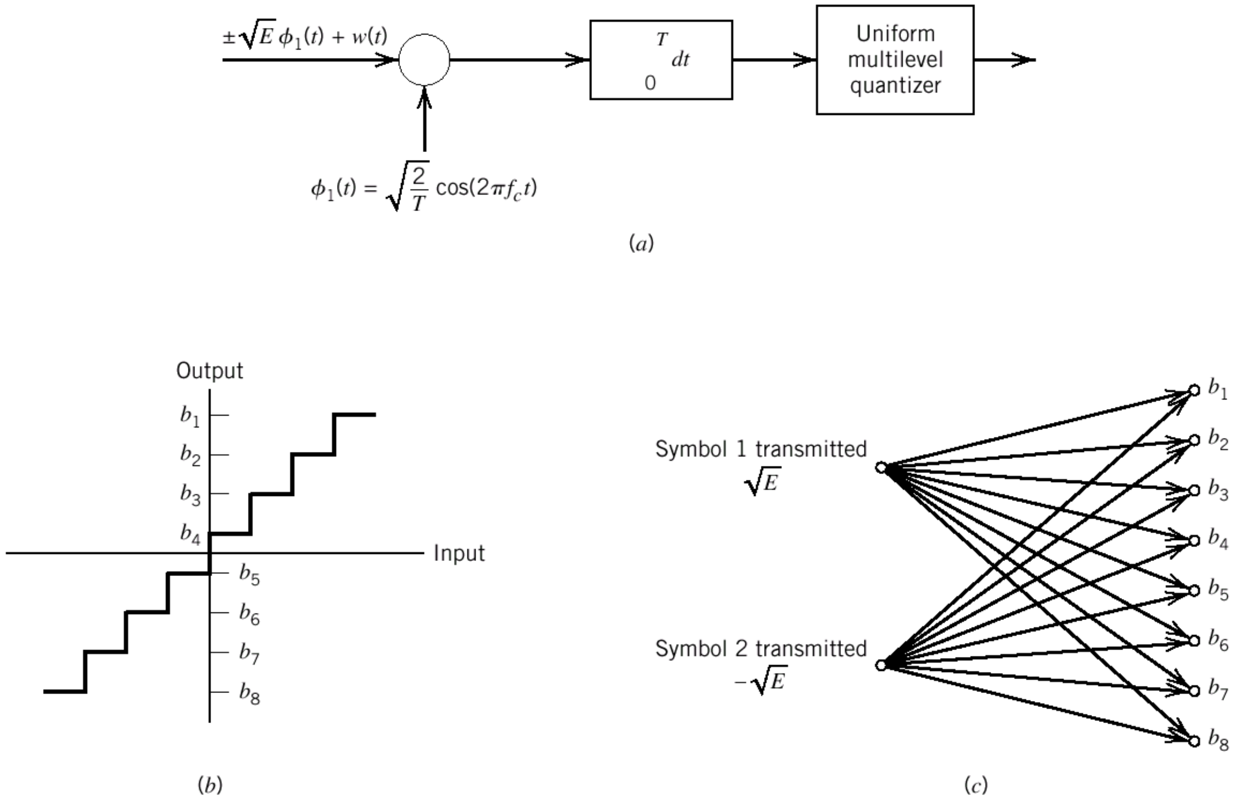


Figure 10.3 Binary input Q-ary output discrete memoryless channel.
(*a*) Receiver for binary phase-shift keying. (*b*) Transfer characteristic of multilevel quantizer. (*c*) Channel transition probability diagram. Parts (*b*) and (*c*) are illustrated for eight levels of quantization.

10.2 Discrete-Memoryless Channels

- Channel coding theorem revisited
  - Channel capacity : maximum amount of information transmitted per channel use (in chapter 9)
  - Channel coding theorem: if R(rate of source information) ≤ C(capacity of a discrete memoryless channel), then there exists a coding technique such that the output of source may be transmitted over the channel with an arbitrarily low probability of symbol error (P_e -> 0.)

10.2 Discrete-Memoryless Channels

- Channel coding theorem revisited (cont.)
  - Channel coding theorem
    - existence of good codes
    - non-constructive
  - error-control coding techniques: provide different methods of designing good codes
  - Notation
    - modulo-2 addition -- EXCLUSIVE OR operation
    - modulo-2 multiplication -- AND operation

10.3 Linear Block Codes

- **Definition of linear code:** if any two code words in the code can be added in modulo-2 arithmetic to produce a third code word in the code. (封闭性)
- **Systematic code:** the message bits are transmitted in unaltered form. It simplifies implementation of the decoder.

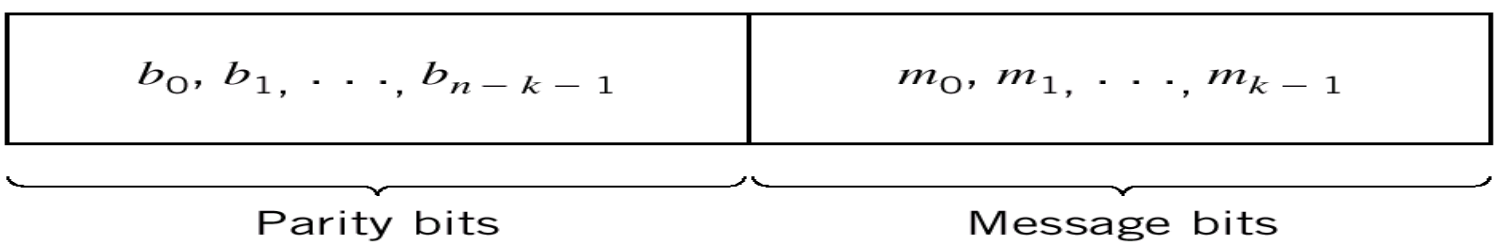


Figure 10.4 Structure of systematic code word.

10.3 Linear Block Codes

- (n, k) linear block code

$c_0, c_1, \ldots, c_{n-1}$: n code bits

$m_0, m_1, \ldots, m_{k-1}$: k message bits

$b_0, b_1, \ldots, b_{n-k-1}$: (n-k) parity bits, which are computed from the message bits according to a prescribed encoding rule

10.3 Linear Block Codes

- Mathematical structure

$$c_i = \begin{cases} b_i & i=0, 1, \ldots, n-k-1 \\ m_{i+k-n} & i=n-k, n-k+1, \ldots, n-1 \end{cases} \qquad (10.1)$$

$$b_i = p_{0i}m_0 + p_{1i}m_1 + \ldots + p_{k-1,\,i}m_{k-1} \qquad (10.2)$$

$$\text{where } p_{ji} = \begin{cases} 1 & \text{if } b_i \text{ depends on } m_j \\ 0 & \text{otherwise} \end{cases} \qquad (10.3)$$

10.3 Linear Block Codes

- Matrix form

$$\mathbf{m} = [m_0, \quad m_1, \quad \ldots, \quad m_{k-1}] \qquad (10.4)$$

$$\mathbf{b} = [b_0, \quad b_1, \quad \ldots, \quad b_{n-k-1}] \qquad (10.5)$$

$$\mathbf{c} = [c_0, \quad c_1, \quad \ldots, \quad c_{n-1}] \qquad (10.6)$$

$$\mathbf{b} = \mathbf{mP} \qquad (10.7)$$

$$\mathbf{P} = \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0,\, n-k-1} \\ p_{10} & p_{11} & \cdots & p_{1,\, n-k-1} \\ & & & \\ p_{k-1,\, 0} & p_{k-1,\, 1} & \cdots & p_{k-1,\, n-k-1} \end{bmatrix} \qquad (10.8)$$

10.3 Linear Block Codes

- **Matrix form** (cont.)

From the equations in (10.4)-(10.6), c may be expressed as follows:

$$c = [\ b \mid m\ ] \quad (10.9)$$

$$= m[P \mid I_k] \quad (10.10)$$

$$= mG \quad (10.13)$$

Where G is the *k-by-n* *generator matrix*

$$G = [P \mid I_k] \quad (10.12)$$

where I_k is the *k-by-k identity matrix* .

G's k rows are linearly independent.

10.3 Linear Block Codes

- Matrix form (cont.)

Let **H** denote the *(n-k)-by-n* *parity-check matrix*

$$\mathrm{H} = [\mathrm{I}_{n-k} \mid \mathrm{P}^T] \tag{10.14}$$

H's (n-k) rows are linearly independent.

We get parity-check equation as follows:

$$\mathrm{cH}^T = \mathrm{mGH}^T = \vec{0}_{1\times(n-k)} \tag{10.16}$$

$$GH^T = [0]_{k\times(n-k)}$$

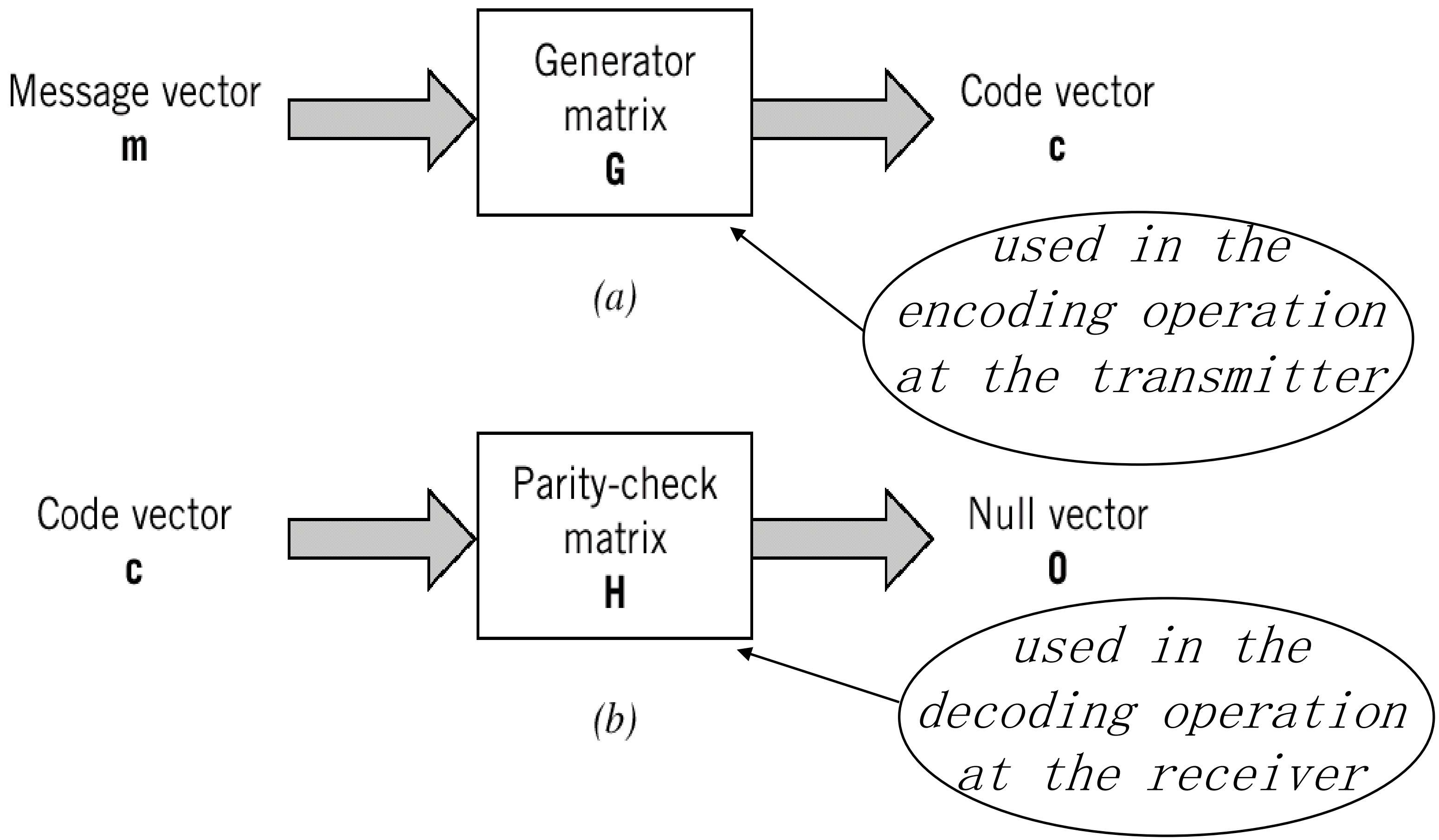


Figure 10.5 Block diagram representations of the generator equation (10.13) and the parity-check equation (10.16).

10.3 Linear Block Codes

- Example 10.1 Repetition Codes
  - (n, 1) block code
  - two code words in the code: all-zero code word & all-one code word
  - For n=5, the **generator matrix** is

$$\mathbf{G} = [\underbrace{\overset{P}{1\ 1\ 1\ 1}}\ |\ \underbrace{\overset{I_1}{1}}]$$

The **parity-check matrix** is

$$\mathbf{H} = \left[\begin{array}{cccc:c} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

I_4

$[I_{n-1} \vdots P^T]$

$[I_4 \vdots P^T]$

10.3 Linear Block Codes

- Syndrome: Definition and Properties

校正子

Let **r** denote the 1-*by-n received vector,* we express the vector **r** as

$$\mathbf{r} = \mathbf{c} + \mathbf{e} \tag{10.17}$$

Where **c** -- the original ***code vector***

e -- the ***error vector*** *or* ***error pattern***.

$$e_i = \begin{cases} 1 & \text{if an error has occurred in the } i\text{th location} \\ 0 & \text{otherwise} \end{cases} \tag{10.18}$$

10.3 Linear Block Codes

- Syndrome (cont.)
  - Definition

    The *error-syndrome vector* (or *syndrome*) is defined as:

$$s = rH^T = (c+e)H^T = \underbrace{cH^T}_{0} + eH^T \qquad (10.19)$$

  - Property 1

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

$$s = eH^T \qquad (10.20)$$

10.3 Linear Block Codes

- Syndrome (cont.)
  - Property 2

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

$\vec{s} \leftrightarrow \{\vec{e}_i,\ i=1,2,\cdots 2^k\}$

We define the 2^k distinct vectors $\mathbf{e}_i$ *as*

$\vec{e}_i = \vec{e}_j + \vec{c}_k$

$$\mathbf{e}_i = \mathbf{e} + \mathbf{c}_i \qquad (10.21)$$

We get $$\mathbf{e}_i\mathbf{H}^T = \mathbf{e}\mathbf{H}^T \qquad (10.22)$$

陪集

Coset of the code -- the set of vectors $\{\mathbf{e}_i,\ i=0, 1, \ldots, 2^k-1\}$

10.3 Linear Block Codes

- Syndrome (cont.)

From Equ.(10.22), we get

$$\mathbf{s} = \mathbf{e}\mathbf{H}^T = \mathbf{e}_i\mathbf{H}^T$$

So the information contained in the syndrome **s** about the error pattern **e** is *not enough* for the decoder to compute the **exact value** of the transmitted code vector. Nevertheless, knowledge of the syndrome s reduces the search for the true error pattern e from 2^n *to* 2^{n-k} possibilities.

10.3 Linear Block Codes

- Minimum distance considerations
  - *Hamming distance* $d(c_1, c_2)$ *-- the number of locations in which their respective elements differ*
  - *Hamming weight* $w(c)$ *-- the number of nonzero elements in the code vector*
  - *minimum distance* d_{min} *-- the smallest hamming distance between any pair of code vectors in the code*

    minimum distance d_{min} *= smallest Hamming weight of the nonzero code vectors*

10.3 Linear Block Codes

- Minimum distance considerations (cont.)
  - *relation between the minimum distance d_{min} and the parity-check matrix H*

    Express the matrix **H** in terms of its columns

    $$H = [h_1, h_2, ..., h_n]$$

    And a code vector **c** satisfy the equation

    $$Hc^T = 0 \quad \rightarrow d_{min} \leq n-k+1$$

  Hence, *the minimum distance of a linear block code is defined by the minimum number of columns of the matrix* H(or rows of the matrix H^T) *whose sum is equal to the zero vector.*

10.3 Linear Block Codes

- *relation between the minimum distance d_{min} and the error-correcting capability of the code*
  - strategy for the decoder -- pick the code vector closest to the received vector **r**.
  - Detect all error patterns of Hamming weight $w(\mathbf{e}) \leqslant t_1$

    if and only if $d_{min} \geqslant t_1 + 1$
  - Correct all error patterns of Hamming weight $w(\mathbf{e}) \leqslant t_2$

    if and only if $d_{min} \geqslant 2t_2 + 1$
  - Detect all error patterns of Hamming weight $w(\mathbf{e}) \leqslant t_1$, and correct all error patterns of Hamming weight $w(\mathbf{e}) \leqslant t_2$

    if and only if $d_{min} \geqslant t_1 + t_2 + 1 \quad (t_1 > t_2)$

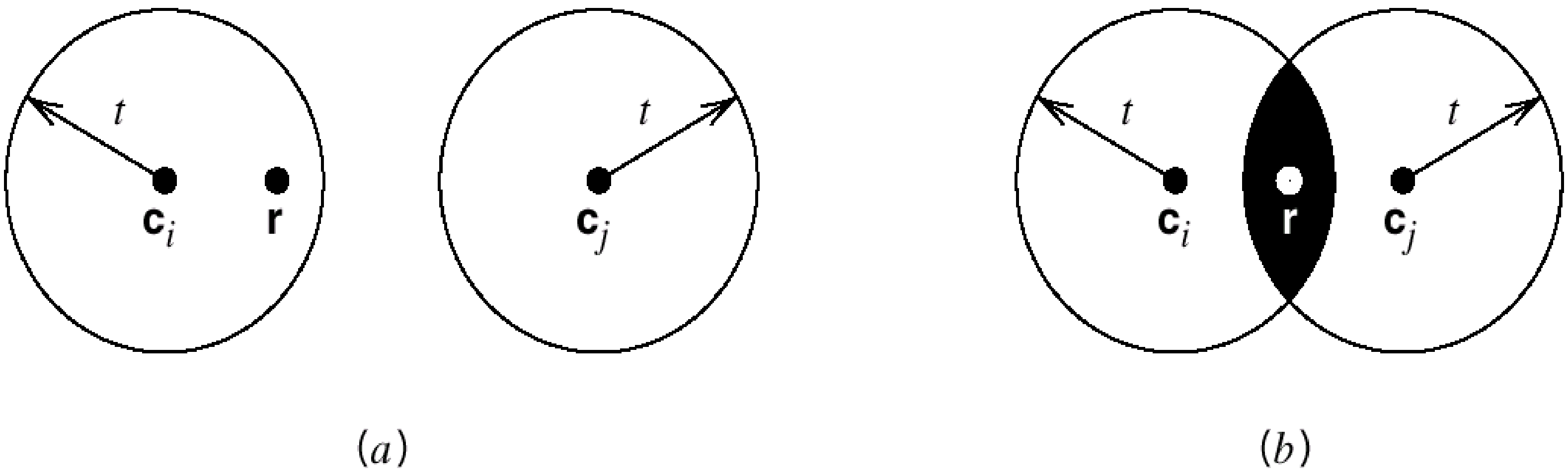


Figure 10.6

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

10.3 Linear Block Codes

- Syndrome decoding
  - 1. compute the syndrome $s = rH^T$
  - 2. according to s, identify the coset leader, call it e_0
  - 3. compute the code vector

$$c = r + e_0$$

as the decoded version of the received vector r

Note: each coset leader has the minimum Hamming weight in its coset

陪集首

Coset leader

$$\begin{array}{ccccccc}
\mathbf{c}_1 = \mathbf{0} & \mathbf{c}_2 & \mathbf{c}_3 & \cdots & \mathbf{c}_i & \cdots & \mathbf{c}_{2^k} \\
\mathbf{e}_2 & \mathbf{c}_2 + \mathbf{e}_2 & \mathbf{c}_3 + \mathbf{e}_2 & \cdots & \mathbf{c}_i + \mathbf{e}_2 & \cdots & \mathbf{c}_{2^k} + \mathbf{e}_2 \\
\mathbf{e}_3 & \mathbf{c}_2 + \mathbf{e}_3 & \mathbf{c}_3 + \mathbf{e}_3 & \cdots & \mathbf{c}_i + \mathbf{e}_3 & \cdots & \mathbf{c}_{2^k} + \mathbf{e}_3 \\
\vdots & \vdots & \vdots & & \vdots & & \vdots \\
\mathbf{e}_j & \mathbf{c}_2 + \mathbf{e}_j & \mathbf{c}_3 + \mathbf{e}_j & \cdots & \mathbf{c}_i + \mathbf{e}_j & \cdots & \mathbf{c}_{2^k} + \mathbf{e}_j \\
\vdots & \vdots & \vdots & & \vdots & & \vdots \\
\mathbf{e}_{2^{n-k}} & \mathbf{c}_2 + \mathbf{e}_{2^{n-k}} & \mathbf{c}_3 + \mathbf{e}_{2^{n-k}} & & \mathbf{c}_i + \mathbf{e}_{2^{n-k}} & & \mathbf{c}_{2^k} + \mathbf{e}_{2^{n-k}}
\end{array}$$

coset

Subset D_1 D_2 D_{2k}

Figure 10.7

Standard array for an (n, k) block code.

10.3 Linear Block Codes

- Example 10.2 Hamming Codes
  - Block length : $n = 2^m - 1 \ (m \geqslant 3)$
    Number of message bits: $k = 2^m - 1 - m$
    Number of parity bits: $m = n - k$
    minimum distance: $d_{min} = 3$
  - generator matrix
  - parity-check matrix
  - relation between the minimum distance d_{min} and the parity-check matrix H
  - syndrome decoding

10.3 Linear Block Codes

•Dual Code

Every (n, k) linear block code with *generator matrix* **G** and *parity-check matrix* **H** has a ***dual code*** with parameters $(n, n-k)$, *generator matrix* **H** and *parity-check matrix* **G**.

10.4 Cyclic Codes

- a subclass of linear block codes
- easy to encode
- possess a well-defined mathematical structure
- efficient decoding schemes
- two fundamental properties:
  - *linearity property*
  - *cyclic property*

10.4 Cyclic Codes

- *Code polynomial*

$$c(X) = c_0 + c_1 X + c_2 X^2 + \dots + c_{n-1} X^{n-1} \qquad (10.27)$$

where X is an indeterminate. For binary codes, the coefficients c_i are 1s and 0s. Multiplication of the polynomial $c(X)$ by X may be viewed as a shift to the right.

- *Lemma: If $c(X)$ is a cyclic code polynomial, then the polynomial*

$$c^{(i)}(X) = X^i c(X) \ \ mod\ (X^n + 1) \qquad (10.33)$$

is also a code polynomial for any cyclic shift i.

10.4 Cyclic Codes

- *Generator polynomial*

$$g(X) = 1 + g_1 X + g_2 X^2 + \ldots + g_{n-k-1} X^{n-k-1} + X^{n-k} \tag{10.34}$$

where the coefficients g_i is equal to 1 or 0.

Note:

- *A cyclic code is uniquely determined by the generator polynomial $g(X)$.*
- *$g(X)$ is a polynomial of degree $(n-k)$ (the polynomial of least degree in the code).*
- *$g(X)$ is a factor of $(X^n + 1)$.*

10.4 Cyclic Codes

- *Encoding procedure for an (n, k) systematic cyclic code*
  - 1.Multiply the message polynomial $m(X)$ by X^{n-k}.

$$m(X) = m_0 + m_1 X + \ldots + m_{k-1} X^{k-1} \qquad (10.36)$$

  - 2.Divide $X^{n-k} m(X)$ by the generator polynomial $g(X)$, obtaining the remainder $b(X)$.
  - 3.Add $b(X)$ to $X^{n-k} m(X)$, obtaining the code polynomial $c(X)$.

$$c(X) = b(X) + X^{n-k} m(X)$$

10.4 Cyclic Codes

- *Parity-check Polynomial*

$$h(X) = 1 + h_1 X + h_2 X^2 + \cdots + h_{k-1} X^{k-1} + X^k \qquad (10.40)$$

where the coefficients h_i are 1 or 0.

Note:

- *A cyclic code is also uniquely specified by the generator polynomial $h(X)$.*
- *$h(X)$ is a polynomial of degree k.*
- *$h(X)$ is also a factor of $(X^n + 1)$ and it satisfies*

$$g(X)h(X) = X^n + 1 \qquad (10.42)$$

10.4 Cyclic Codes

- *Generator and Parity-check Matrices*
  - *Generator matrix*

$$G(X)=\begin{bmatrix} g(X) \\ Xg(X) \\ \vdots \\ X^{k-1}g(X) \end{bmatrix}$$

  - *Parity-check matrix*

$$H(X)=\begin{bmatrix} X^{k}h(X^{-1}) \\ X^{k+1}h(X^{-1}) \\ \vdots \\ X^{n-1}h(X^{-1}) \end{bmatrix}$$

10.4 Cyclic Codes

- *Encoder for cyclic codes*
  - *Encoding procedure for an (n, k) systematic cyclic code*

1. multiplication of the message polynomial $m(X)$ *by* X^{n-k}
2. division of $X^{n-k}m(X)$ by the generator polynomial $g(X)$ *to* obtain the remainder $b(X)$
3. addition of $b(X)$ to $X^{n-k}m(X)$

These three steps can be implemented by means of the encoder shown in Fig. 10.8, consisting of a *linear feedback shift register* with *(n-k)* stages.

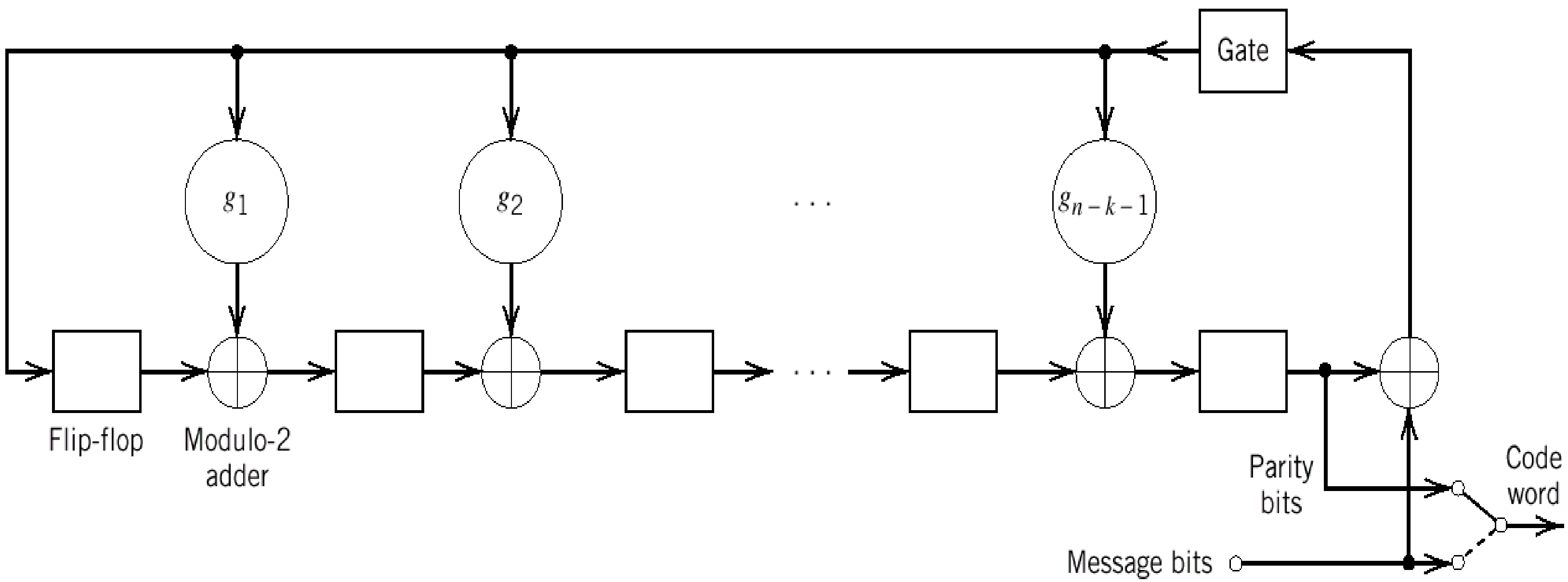


Figure 10.8
Encoder for an (n, k) cyclic code.

10.4 Cyclic Codes

- *Encoder for cyclic codes* (*cont.*)

The operation of the encoder shown in Fig. 10.8 proceeds as follows:

- The gate is switched on. Hence, the k message bits are shifted into the channel. As soon as the k message bits have entered the shift register, the resulting $(n-k)$ bits in the register form the parity bits.
- The gate is switched off, thereby breaking the feedback connections.
- The contents of the shift register are read out into the channel.

10.4 Cyclic Codes

- *Calculation of the syndrome*

Let the received word be represented by a polynomial as follows:

$$r(X) = r_0 + r_1 X + \ \ldots \ + r_{n-1} X^{n-1} \qquad (10.46)$$

and $r(X)$ may be expressed as:

$$r(X) = q(X)g(X) + s(X) \qquad (10.47)$$

where $q(X)$ denotes the quotient and $s(X)$ denotes the *syndrome polynomial* (remainder, degree $\leqslant n-k-1$).

Fig.10.9 shows a *syndrome calculator*. (identical to the encoder of Fig.10.8)

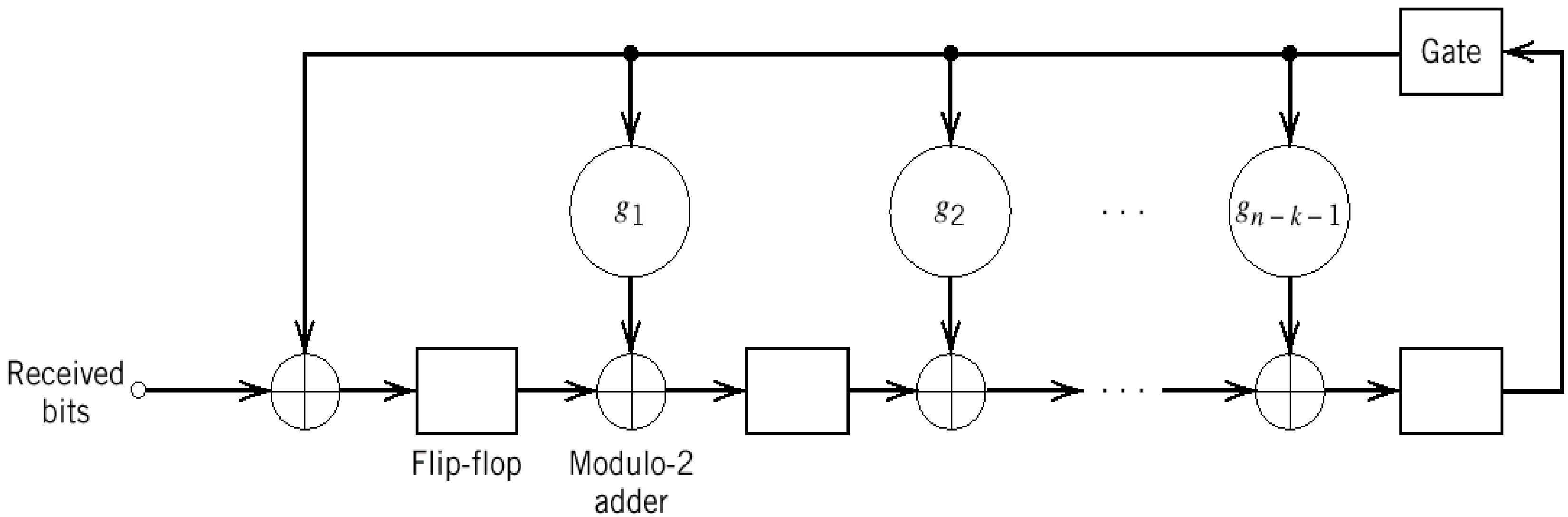


Figure 10.9
Syndrome calculator for (n, k) cyclic code.

10.4 Cyclic Codes

- *Properties of the syndrome polynomial*
  1. *The syndrome of a received word polynomial is also the syndrome of the corresponding error polynomial.*
  2. *Let $s(X)$ be the syndrome of a received word polynomial $r(X)$. Then, the syndrome of $Xr(X)$, a cyclic of $r(X)$, is $Xs(X)$ (mod $g(X)$).*
  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 $r(X)$.*

10.4 Cyclic Codes

- *Example 10.3 Hamming codes revisited*

  (7.4) cyclic code
  - generator polynomial
  - parity-check polynomial
  - construction of a code word in systematic form
  - generator matrix
  - parity-check matrix
  - encoder (Fig.10.10)
  - syndrome calculator (Fig.10.11)

*** Any cyclic code generated by a primitive polynomial is a Hamming code of minimum distance 3.**

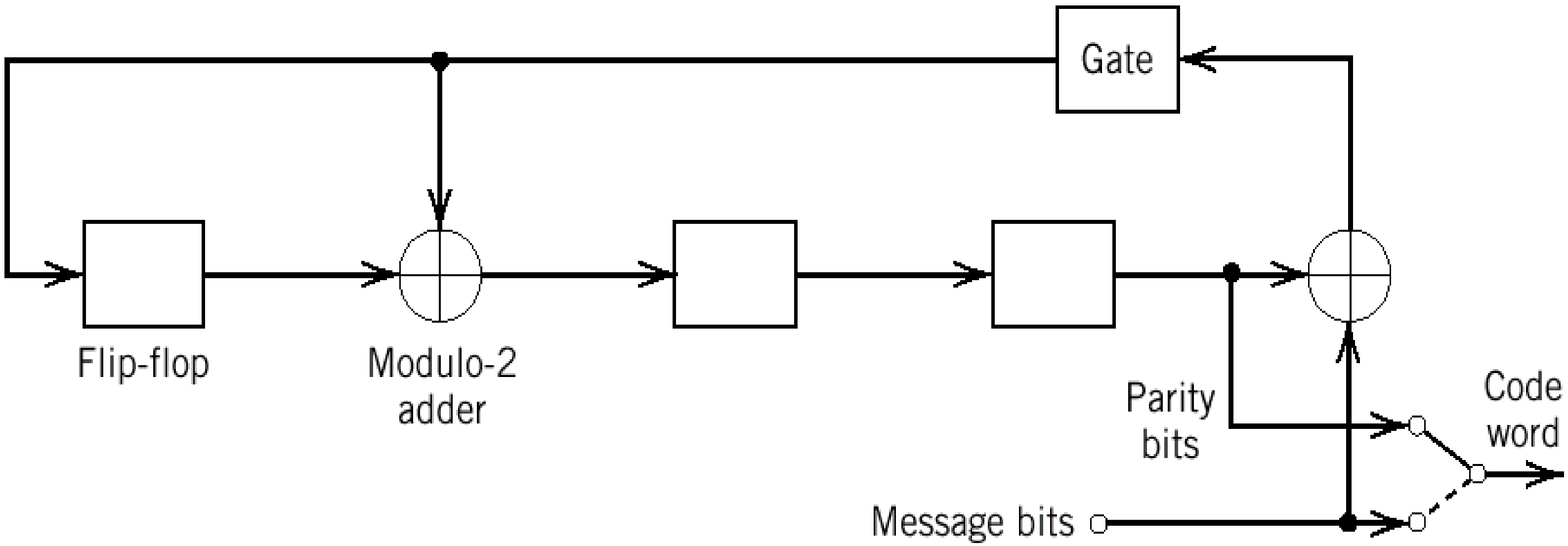


Figure 10.10
Encoder for the (7, 4) cyclic code generated by
$g(X) = 1 + X + X^3$.

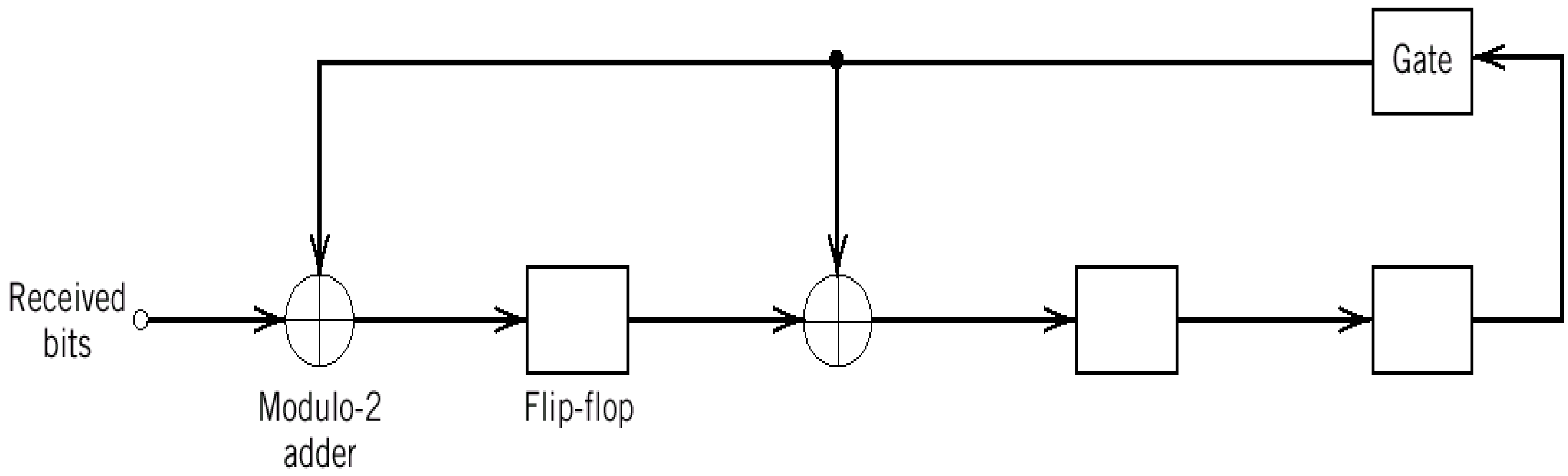


Figure 10.11

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

10.4 Cyclic Codes

- *Example 10.4 Maximal-Length codes*
  - Block length : $n = 2^m - 1 \ (m \geq 3)$
  - Number of message bits: $k = m$
  - Minimum distance: $d_{min} = 2^{m-1}$
  - Generator polynomial

$$g(X) = (X^n + 1)/h(X) \qquad (10.52)$$

where $h(X)$ is any primitive polynomial of degree m.

* Maximal-length codes are the dual of Hamming codes. The polynomial h(X) defines the feedback connections of the encoder. The generator polynomial g(X) defines one period of the maximal-length code, assuming the initial state is 00...01.

10.4 Cyclic Codes

- ***Example 10.4*** ***Maximal-Length codes***

  (7,3) maximal-length code -- dual of the (7,4) Hamming code

  Block length : $n = 7$

  Number of message bits: $k = 3$

  Minimum distance: $d_{min} = 4$

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

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

initial state: 0 0 1

output sequence: 1 1 1 0 1 0 0

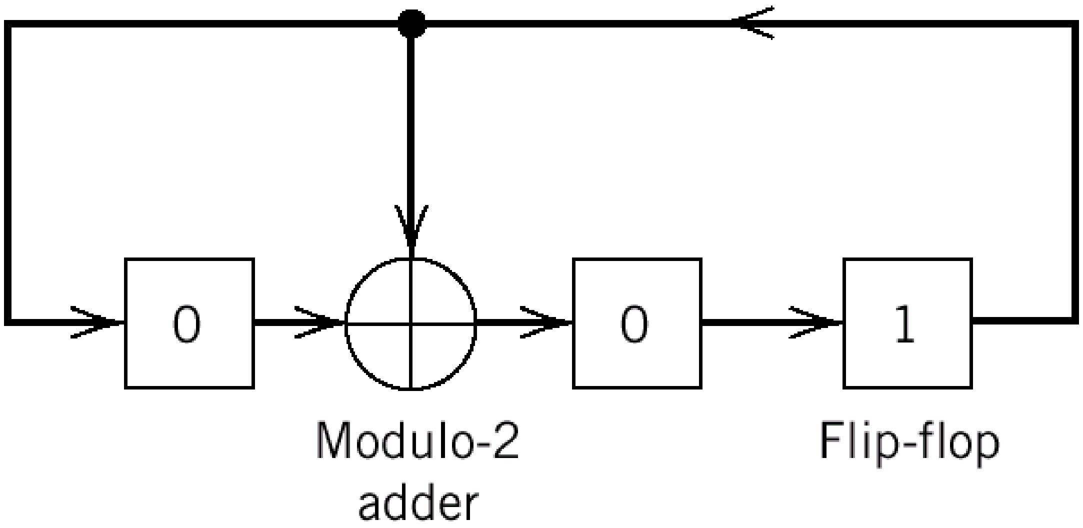


Figure 10.12
Encoder for the (7, 3) maximal-length code; the initial state of the encoder is shown in the figure.

10.4 Cyclic Codes

- *Cyclic redundancy check (CRC) codes*
  - well-suited for *error detection*
    - *can detect many combinations of likely errors*
    - *easy implementation of both encoding and error-detecting*
  - ***commonly used in***
    - *automatic-repeat request(ARQ) strategies*
    - *digital subscriber lines*

10.4 Cyclic Codes

- *Cyclic redundancy check(CRC) codes*
  - ***error patterns that binary (n,k) CRC codes can detect***
    - *All error bursts of length n-k or less.*
    - *A fraction of error bursts of length equal to n-k+1; the fraction equals $1-2^{-(n-k-1)}$.*
    - *A fraction of error bursts of length greater than n-k+1; the fraction equals $1-2^{-(n-k-1)}$.*
    - *All combinations of $d_{min}-1$ (or fewer) errors.*
    - *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.*

10.4 Cyclic Codes

- *Bose-Chaudhuri-Hocquenghem (BCH) codes*
  - **t-error correcting cyclic codes** (can detect & correct up to t random error per code word)
  - **offer flexibility in the choice of code parameters**(block length & code rate)
  - **be among the best known code the same block length and code rate**
  - **Primitive BCH codes**
    - Block length : $n = 2^m - 1 \ (m \geq 3)$
    - Number of message bits: $k \geq n - mt$
    - Minimum distance: $d_{min} \geq 2t + 1$
    - Maximum number of detectable errors: t

10.4 Cyclic Codes

- *Reed-Solomon(RS) codes*
  - *subclass of nonbinary BCH codes*
  - *t-error-correcting RS codes*
    - 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

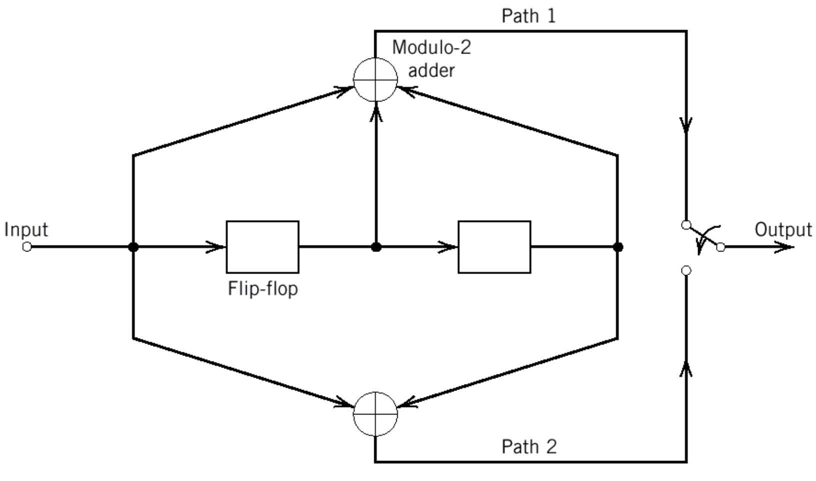
10.4 Cyclic Codes

Reasons for wide application of RS codes

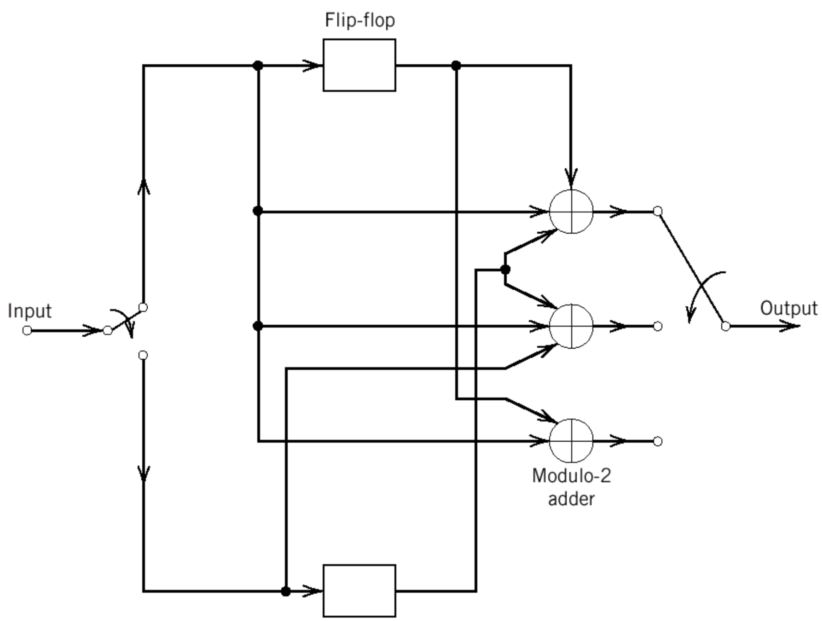
- *make highly efficient use of redundancy*
- *block lengths and symbol sizes can be adjusted to accommodate a wide range of message sizes*
- *provide a wide range of code rates*
- *efficient decoding techniques are available for use*

10.5 Convolutional Codes

- Characteristic:
  - message bits come in *serially* rather than in *large blocks*
  - encoder generates redundant bit by using *modulo-2 convolutions*
- rate $1/n$ convolutional encoder consists of:
  - an M-stage shift register
  - n modulo-2 adders
  - a multiplexer serializes the outputs of the adders
- constraint length $K = M + 1$



(a)



(b)

Figure 10.13

(*a*) Constraint length-3, rate - $\frac{1}{2}$ convolutional encoder.

(*b*) Constraint length-2, rate- $\frac{2}{3}$ convolutional encoder.

*** The use of nonsystematic codes is ordinarily preferred over systematic codes in convolutional coding.**

10.5 Convolutional Codes

- Generator polynomial

$$g^{(i)}(D) = g_0^{(i)} + g_1^{(i)} D + g_2^{(i)} D^2 + \cdots + g_M^{(i)} D^M \qquad (10.55)$$

where D --- the *unit-delay variable*

($g_0^{(i)}, g_1^{(i)}, \ldots, g_M^{(i)}$) --- the *generator sequence*, denote the *impulse response* of the ith path.

Here, the impulse response means the connection between the output and the input of a convolutional encoder.

$$g_j^{(i)} = \begin{cases} 1 & \text{if a connection exists between the } i\text{th output and the } j\text{-stage delay of the input} \\ 0 & \text{otherwise} \end{cases}$$

10.5 Convolutional Codes

- Example 10.5 (Fig. 10.13a)
  - generator polynomial (path1 & path2)
  - message polynomial for sequence (10011)
  - output polynomial (path1 & path2)
  - encoded sequence

Note: The message sequence of length L produces an encoded sequence of length $n(L+K-1)$.

A terminating sequence of $(K-1)$ zeros is appended to the last input bit of the message sequence, in order to restore the shift register to its zero initial state.

10.5 Convolutional Codes

- *Graphical forms to portray the structural properties (or input-output relation) of a convolutional encoder*
  - Code Tree (Fig. 10.14)

  Note: The tree becomes repetitive after the first K branches.
  - Trellis (Fig. 10.15)
  - State Diagram (Fig. 10.16)

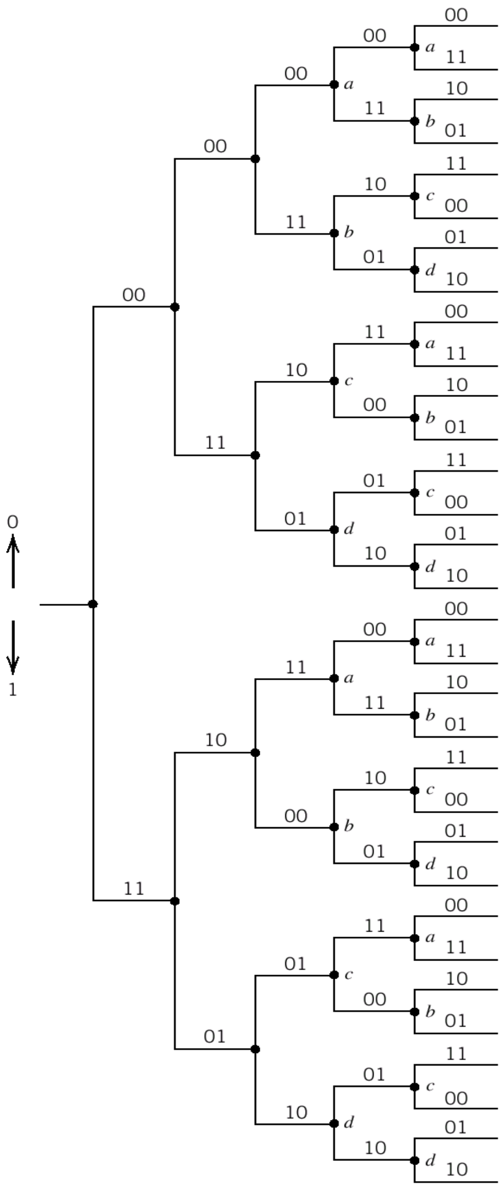


Figure 10.14

Code tree for the convolutional encoder of Figure 10.13*a*.

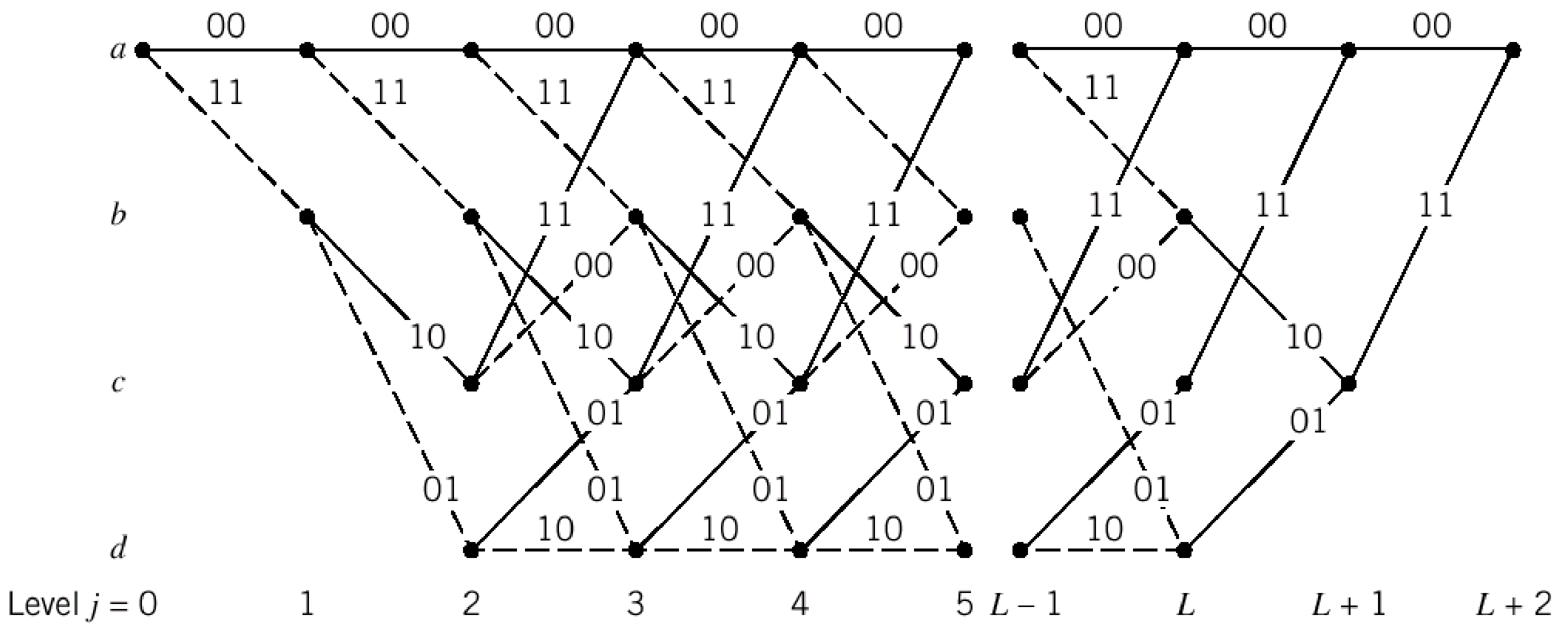


Figure 10.15

Trellis for the convolutional encoder of Figure 10.13*a*.

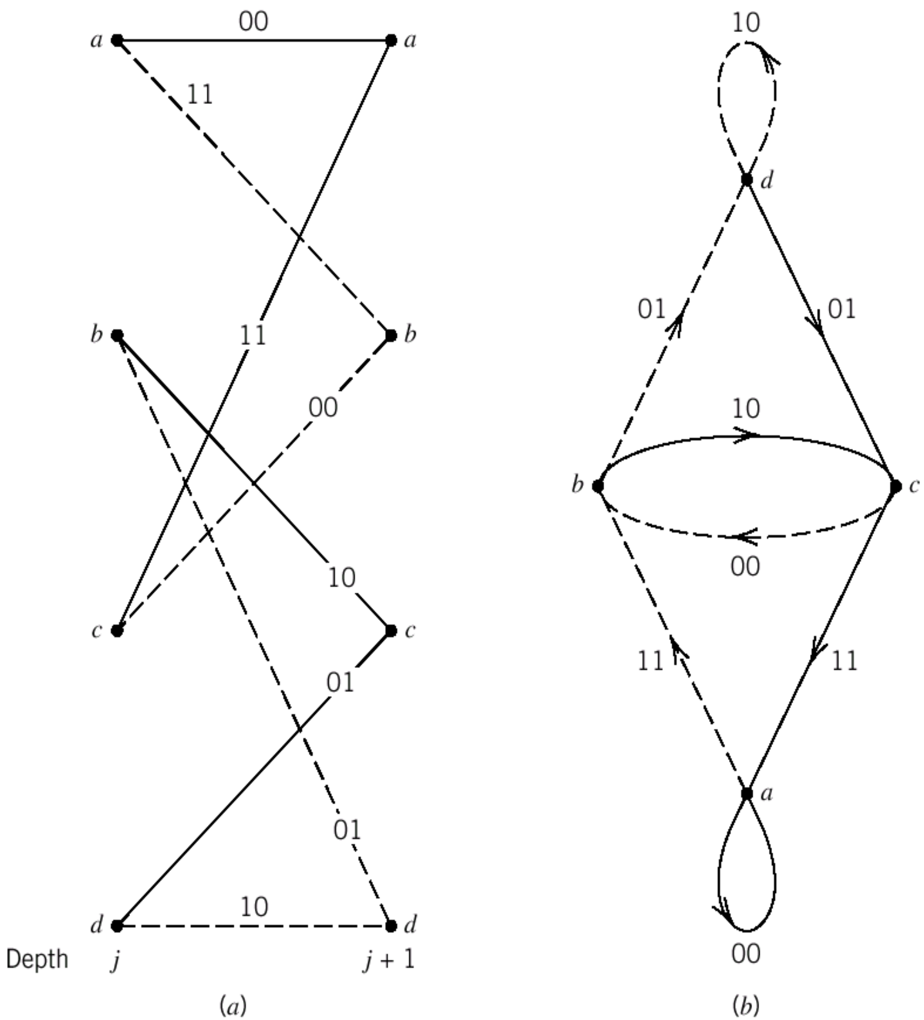


Figure 10.16
(*a*) A portion of the central part of the trellis for the encoder of Figure 10.13*a*. (*b*) State diagram of the convolutional encoder of Figure 10.13*a*.

10.6 Maximum Likelihood Decoding of Convolutional Codes

- Task: **decoding**--Given the *received vector* $\mathbf{r}$, make an *estimate* $\hat{\mathbf{m}}$ of the message vector.
- Method:

$$\underset{\text{(message vector)}}{\mathbf{m}} \xleftrightarrow{\text{one-to-one correspondence}} \underset{\text{(transmitted code vector)}}{\mathbf{c}}$$

put $\hat{\mathbf{m}} = \mathbf{m}$ if and only if $\hat{\mathbf{c}} = \mathbf{c}$

- Decoding rule: *minimize the probability of decoding error(optimum rule).*

10.6 Maximum Likelihood Decoding of Convolutional Codes

Theory of maximum likelihood decoding

For equiprobable messages, the probability of decoding error is minimized if the estimate $\hat{\mathbf{c}}$ *is chosen to maximize the log-likelihood function.*

Decision rule: Choose the estimate $\hat{\mathbf{c}}$ for which the log-likelihood function $\log p(\mathbf{r}|\mathbf{c})$ is maximum. (10.56)

where $p(\mathbf{r}|\mathbf{c})$ -- conditional probability

10.6 Maximum Likelihood Decoding of Convolutional Codes

For binary symmetric channel, r & c are binary sequences of length *N*.

$$p(\mathbf{r}|\mathbf{c}) = \prod_{i=1}^{N} p(r_i | c_i) \qquad (10.57)$$

so

$$\log p(\mathbf{r}|\mathbf{c}) = \sum_{i=1}^{N} \log p(r_i | c_i) \qquad (10.58)$$

where

$$p(r_i/c_i) = \begin{cases} p & \text{if } r_i \neq c_i \\ 1-p & \text{if } r_i = c_i \end{cases} \qquad (10.59)$$

$$\log p(\mathbf{r}|\mathbf{c}) = d \log p + (N-d) \log(1-p)$$

$$= d \log[p/(1-p)] + N \log(1-p) \qquad (10.60)$$

where d is the Hamming distance between r and c.

10.6 Maximum Likelihood Decoding of Convolutional Codes

Maximum likelihood decoding rule for BSC

$$\log p(\mathbf{r}|\mathbf{c}) = d\log[p/(1-p)] + N\log(1-p)$$

For $p<1/2$, $\log[p/(1-p)]$ and $\log(1-p)$ are negative. So the decision rule is:

Choose the estimate $\hat{\mathbf{c}}$ that minimizes the Hamming distance between the received vector r and the transmitted vector c.

(10.61)

maximum likelihood decoder → minimum distance decoder

10.6.1 The Viterbi Algorithm

The Viterbi algorithm – maximum likelihood sequence estimator (MLSE)

It is an efficient algorithm for practical implementation of the maximum likelihood decoding. We may decode a convolutional code by choosing a path in the code trellis whose coded sequence differs from the received sequence in the fewest number of places.

10.6.1 The Viterbi Algorithm

Code trellis for (n,k,K) convolutional code

- $2^{k(K-1)}$ *nodes in the trellis*
- 2^k *branches departing each node in the trellis*
- *at level* $j \geq K$, 2^k *paths entering any of the nodes in the trellis*

10.6.1 The Viterbi Algorithm

The Viterbi algorithm

- ***Initialization***

  Label the left-most state of the trellis (i.e., the all-zero state at level 0) as 0.

- ***Computation step j+1***

  Let j=0,1,2,...,and suppose that at the previous step j we have done two things:

  – All survivor paths are identified.

  – The survivor path and its metric for each state of the trellis are stored.

10.6.1 The Viterbi Algorithm

The Viterbi algorithm (cont.)

Then,at level j+1,

- compute the metric for all the paths entering each state of the trellis by adding the metric of the incoming branches to the metric of the connecting survivor path form level j
- for each state, identify the path with the lowest metric as the survivor of step j+1,thereby updating the computation

- ***Final step***

Continue the computation until the algorithm completes its forward search and reaches the termination node(i.e.,all-zero state).

10.6.1 The Viterbi Algorithm

Decoding window

*When the received sequence is very long, a **decoding window** of length l is specified, and the algorithm operates on a corresponding frame of the received sequence, always stopping after l steps.A decision is then made on the "best" path and the symbol associated with the branch on that path is released to user. Next, the decoding window is moved forward one time interval, and a decision on the next code frame is made, and so on.*

10.6.1 The Viterbi Algorithm

Example 10.6

Correct Decoding of Received All-Zero Sequence

Encoder : Fig. 10.13a

Transmitted sequence: all-zero sequence

Received sequence: (0100010000...)

Decoder(Viterbi algorithm): Fig. 10.17

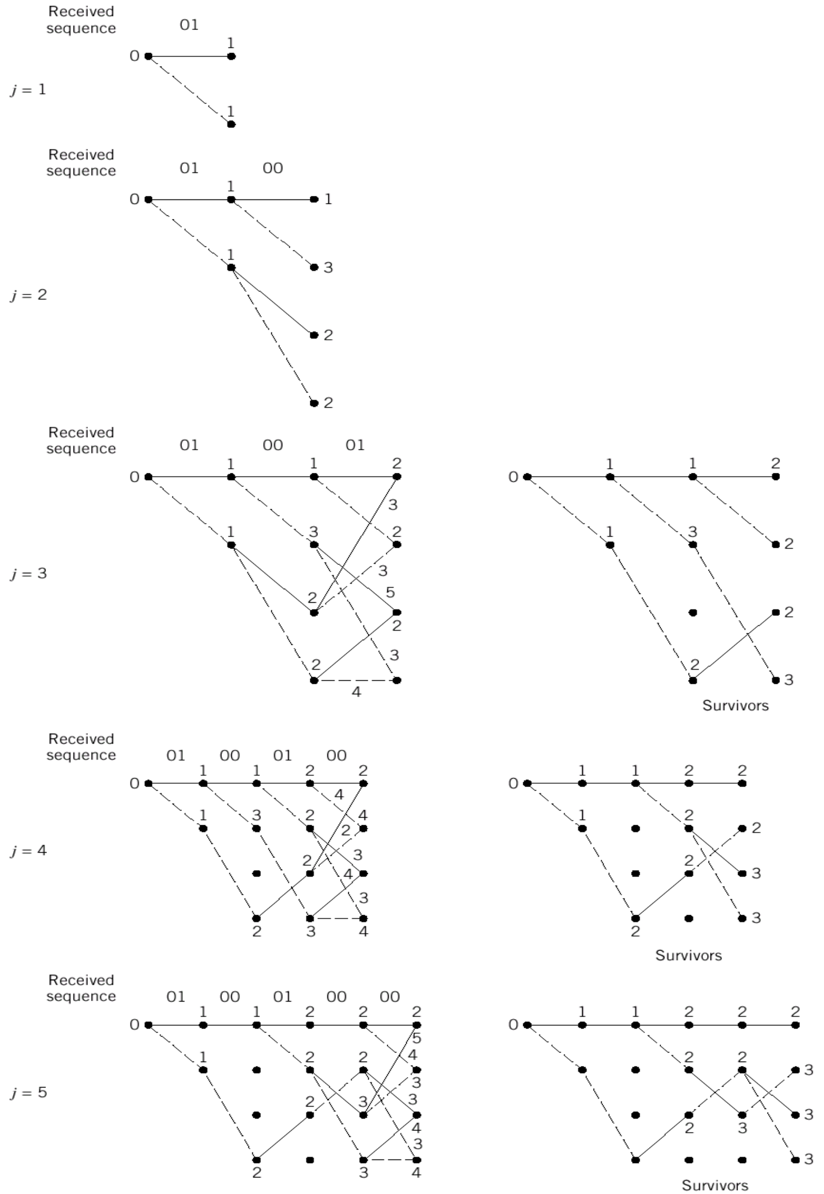


Figure 10.17 Illustrating steps in the Viterbi algorithm for Example 10.6.

10.6.1 The Viterbi Algorithm

Example 10.7

Incorrect Decoding of Received All-Zero Sequence

Encoder : Fig. 10.13a

Transmitted sequence: all-zero sequence

Received sequence: (1100010000...)

Decoder(Viterbi algorithm): Fig. 10.18

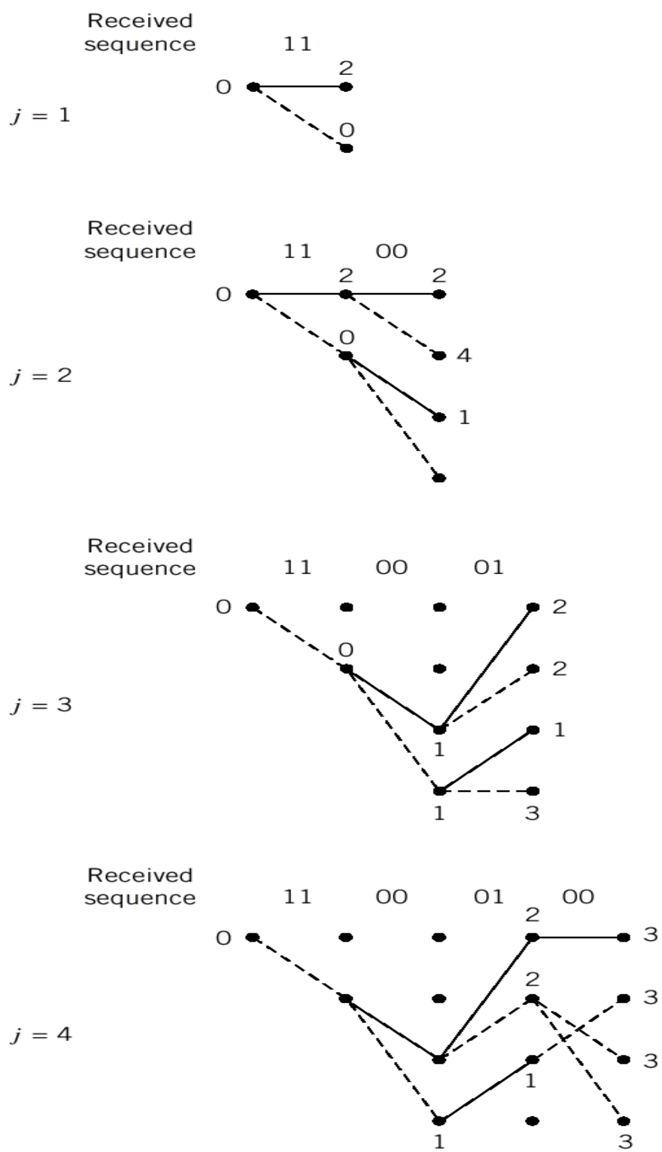


Figure 10.18
Illustrating breakdown of the Viterbi algorithm in Example 10.7.

10.6.2 Free Distance of a Convolutional Code

Why?

The free distance is the most important single measure of a convolutional code's ability to combat channel noise.

Definition

*The free distance (d_{free}) is defined as the **minimum Hamming distance** between any two code words in the code. It can be obtained quite simply from the state diagram of the convolutional encoder.*

10.6.2 Free Distance of a Convolutional Code

Consider an example

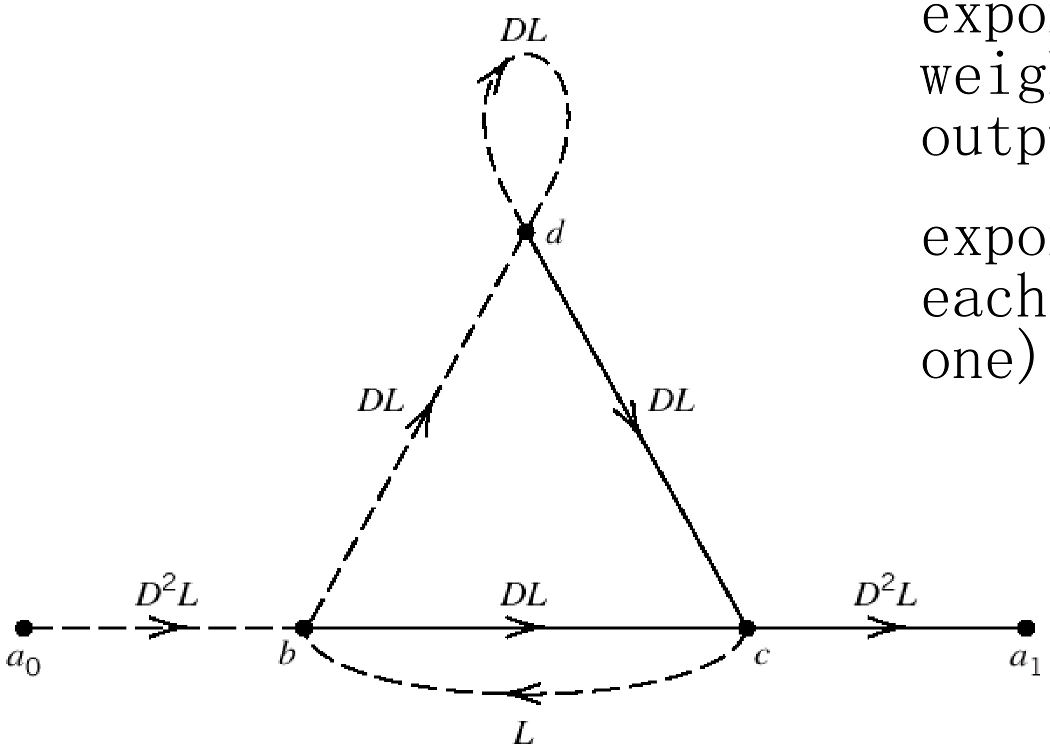
encoder: Fig.10.13a

state diagram: Fig.10.16b

signal-flow graph(modified state diagram:) Fig.10.19

signal-flow graph consists of:

- *a single input & a single output*
- *nodes & directed branches*



D ,L -- dummy variables

exponent of D -- Hamming weight of the encoder output

exponent of L -- length of each branch(always equal to one)

Figure 10.19

Modified state diagram of convolutional encoder.

10.6.2 Free Distance of a Convolutional Code

Operating rules of a signal-flow graph

- A branch multiplies the signal at its input node by the *transmittance* characterizing that branch.
- A node with incoming branches *sums* the signals produced by all of those branches.
- The signal at a node is applied equally to all the branches outgoing from that node.
- The *transfer function* of the graph is the ratio of the output signal to the input signal.

10.6.2 Free Distance of a Convolutional Code

Input-output relations in Fig.10.19

$$b = D^2 L a_0 + Lc$$

$$c = DLb + DLd$$

$$d = DLb + DLd$$

$$a_1 = D^2 Lc \qquad (10.62)$$

Transfer function a_1/a_0

$$T(D,L) = D^5L^3/(1-DL(1+L)) \qquad (10.63)$$

$$= D^5L^3 + D^6L^4 + D^6L^5 + D^7L^5 + 2D^7L^6 + \ldots$$

distance transfer function:

$$T(D,1) = D^5 + 2D^6 + 4D^7 + \ldots \qquad (10.65)$$

10.6.2 Free Distance of a Convolutional Code

Catastrophic code

When *T(D,1)* is nonconvergent, *an infinite number of decoding errors* are caused by a *finite number of transmission errors*; the convolutional code is then subject to catastrophic error propagation, and the code is called a *catastrophic code.*

A systematic convolutional code cannot be catastrophic. But for a prescribed constraint length, it usually has *smaller free distances* than that of nonsystematic convolutional codes.

10.6.3 Asymptotic Coding Gain

Binary symmetric channel

model: BPSK modulation + AWGN channel + hard-decision demodulation

uncoded system : $p_e \propto \exp(-E_b/N_0)$

coded system: $p_e \propto \exp(-d_{free} r E_b/2N_0)$

asymptotic coding gain:

$$G_a = 10\log_{10}(d_{free} r/2) \text{ dB} \qquad (10.66)$$

Where r is the code rate.

10.6.3 Asymptotic Coding Gain

Binary-input AWGN channel

model: BPSK modulation + AWGN channel + no output quantization demodulation

uncoded system : $p_e \propto \exp(-E_b/N_0)$

coded system: $p_e \propto \exp(-d_{free} r E_b/N_0)$

asymptotic coding gain:

$$G_a = 10\log_{10}(d_{free} r) \text{ dB} \qquad (10.67)$$

10.6.3 Asymptotic Coding Gain

hard-decision decoder Equ.(10.66)

soft-decision decoder Equ.(10.67)

unquantized demodulator output instead of making hard decisions → an ***advantage (3dB) gained***

↓

complexity due to the need for accepting analog inputs

We may avoid the for an analog decoder by using a that performs finite output quantization, and yet realize a performance close to the optimum.

10.7 Trellis-Coded Modulation

Problem in the traditional approach to channel coding

Encoding(decoding) is performed *separately* from modulation(detection) in the transmitter (receiver).

transmitting additional redundant bits

error control

increased power efficiency

lowering the information bit rate per channel bandwidth

decreased bandwidth efficiency

10.7 Trellis-Coded Modulation

- Solution

Combine coding and modulation as a single entity to attain a more effective utilization of the available bandwidth and power.

The combination is referred to as trellis-coded modulation(TCM).

10.7 Trellis-Coded Modulation

- **TCM has three basic features:**
  1. The number of signal points in the constellation used is *larger* than what is required for the modulation format of interest with the same data rate; the *additional points allow redundancy* for forward err-control coding *without sacrificing bandwidth*.
  2. Convolutional coding is used to introduce a certain dependency between successive signal points, such that only certain *patterns or sequences of signal points* are permitted.

10.7 Trellis-Coded Modulation

- **TCM has three basic features: (cont.)**

  3. Soft-decision decoding is performed in the receiver, in which the permissible sequence of signals is modeled as a trellis structure; hence, the name ″trellis codes.″

- the size of the constellation ↑---> the probability of symbol error ↑ (for a fixed SNR) ---> soft-decision

10.7 Trellis-Coded Modulation

In AWGN channel, maximum likelihood decoding of trellis codes consists of finding the particular path through the trellis with *minimum squared Euclidean distance* to the received sequence.

Maximizing the Hamming distance ≠ maximizing the squared Euclidean distance **(except for BPSK and QPSK)**

In the design of trellis codes, the emphasis is on *maximizing the Euclidean distance* between code vectors.

10.7 Trellis-Coded Modulation

- **Approach to design the trellis code --** ***set partitioning***

  Partition an M-ary constellation of interest successively into 2, 4, 8, . . . *subsets* with size $M/2, M/4, M/8, \ldots$, and having *progressively larger increasing minimum Euclidean distance* between their respective signal points.

- ***Example of the partitioning procedure (Fig. 10.20 and Fig. 10.21)***

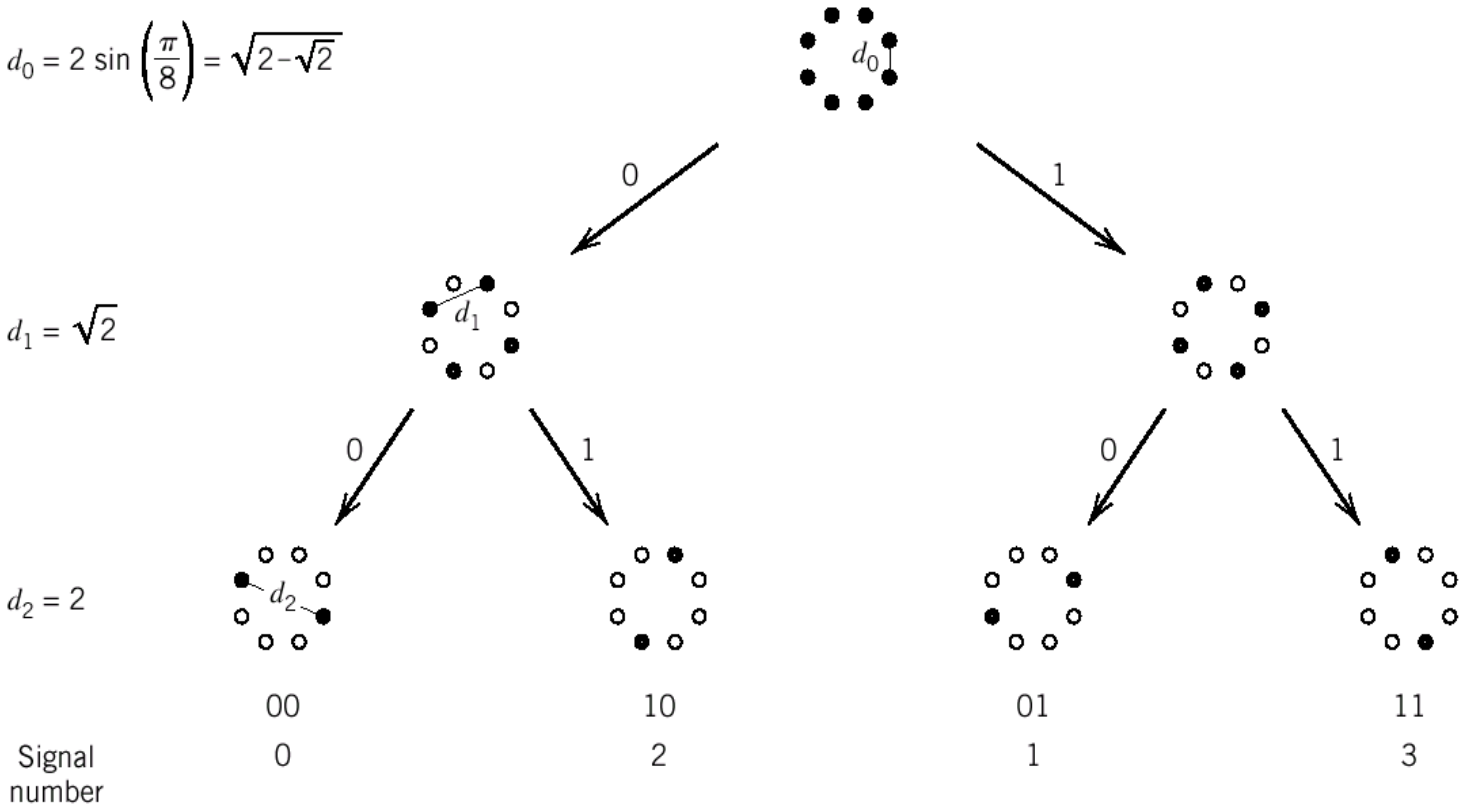


Figure 10.20

Partitioning of 8-PSK constellation(*circular*), which shows that $d_0 < d_1 < d_2$.

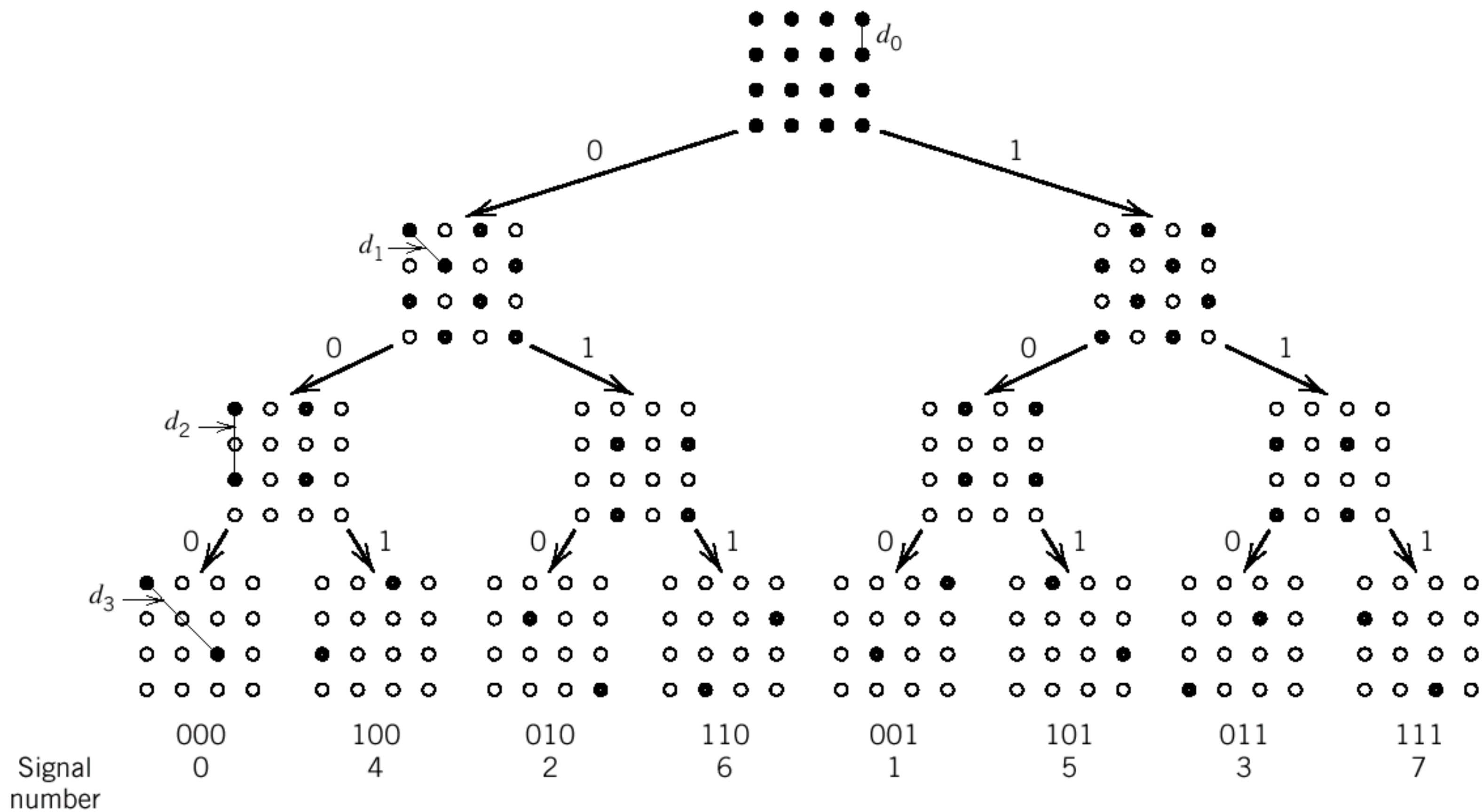


Figure 10.21

Partitioning of 16-QAM constellation *(rectangular)*, which shows that $d_0 < d_1 < d_2 < d_3$.

10.7 Trellis-Coded Modulation

- Ungerboeck codes
  - at transmitter

    1. send n bits/symbol with quadrature modulation
    2. 2-dimensional constellation of 2^{n+1} signal points for modulation
  - at receiver

    Viterbi algorithm used to perform maximum likelihood sequence estimation
  - two examples (Fig.10.22 & Fig.10.23)

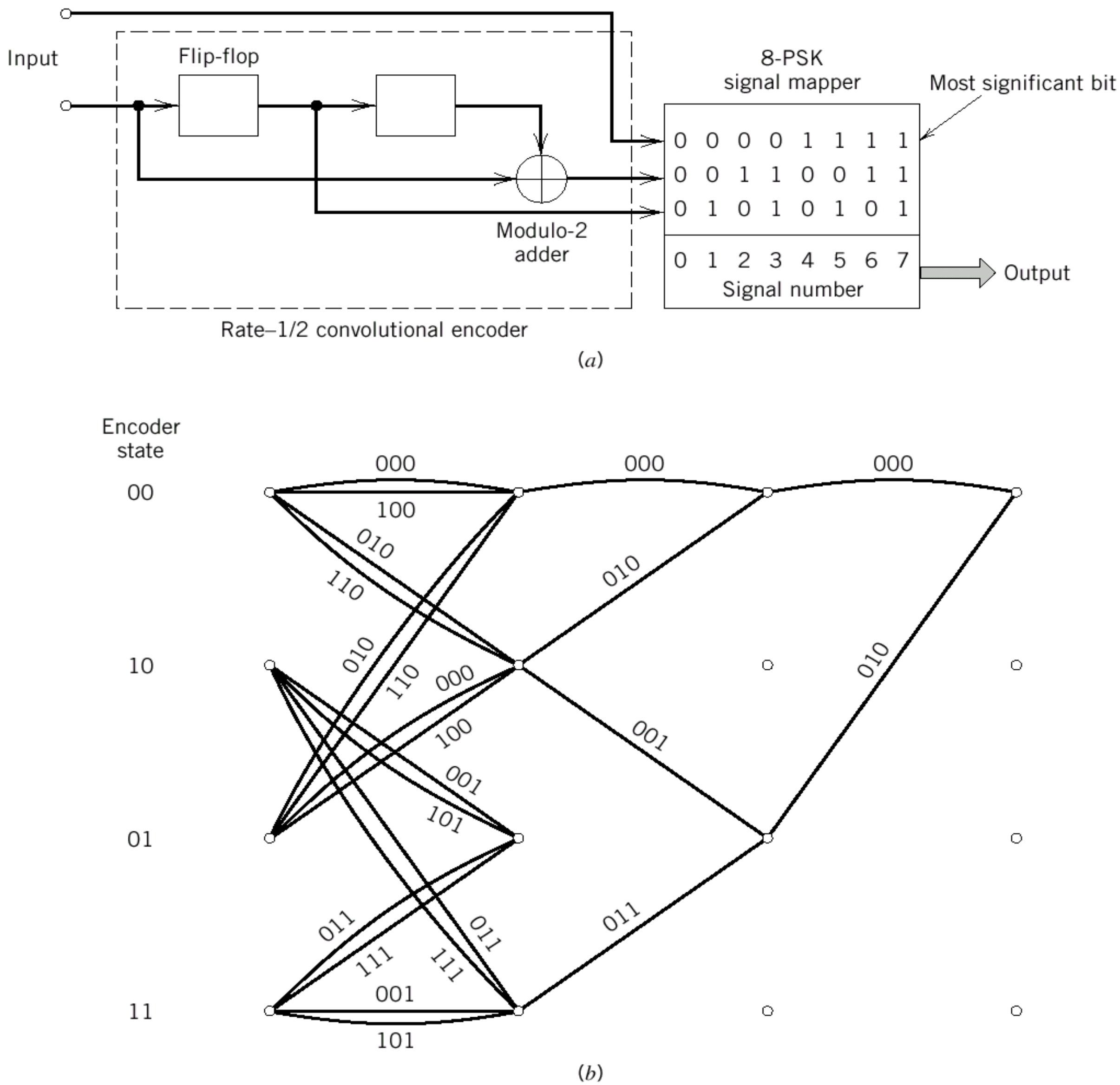
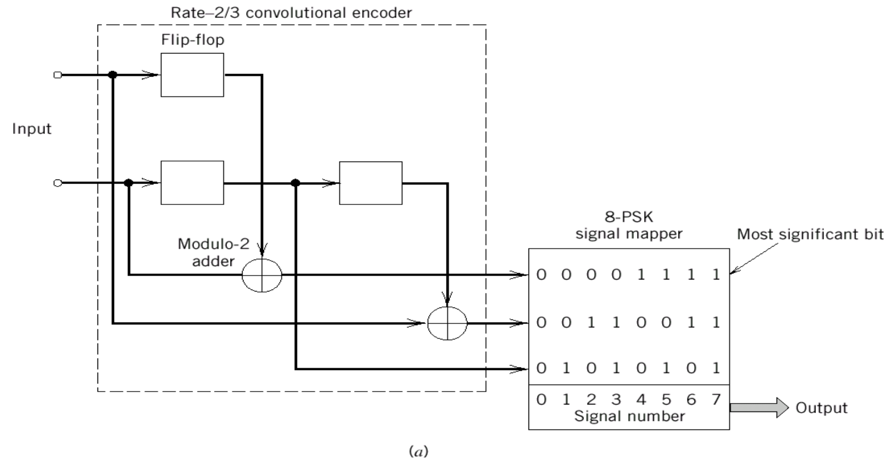


Figure 10.22
(*a*) Four-state Ungerboeck code for 8-PSK; the mapper follows Figure 10.20. (*b*) Trellis of the code.



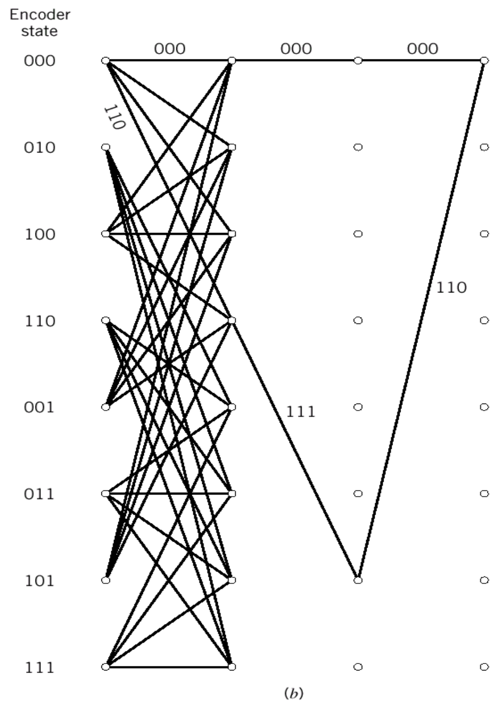


Figure 10.23 (*a*) Eight-state Ungerboeck code for 8-PSK; the mapper follows Figure 10.20. (*b*) Trellis of the code with only some for the branches shown.

10.7 Trellis-Coded Modulation

- Asymptotic coding gain

Definition:

$$G_a = 10\log_{10}(d^2_{free}/\ d^2_{ref}) \qquad (10.68)$$

where d_{free} is the *free Euclidean distance* of the code and d_{ref} is the *minimum Euclidean distance* of an uncoded modulation scheme operating with the same signal energy per bit.

Note: number of states ↑ --> coding gain ↑ (table 10.9)

10.7 Trellis-Coded Modulation

- Asymptotic coding gain(cont.)

Example: (*Fig. 10.24*)

Ungerboeck 8-PSK code of Fig. 10.22a

reference: uncoded 4-PSK

the free Euclidean distance:

$$d_{free} = d_2 = 2$$

the minimum Euclidean distance

$$d_{ref} = \sqrt{2}$$

asymptotic coding gain $G_a = 10\log_{10}2 = 3\text{dB}$

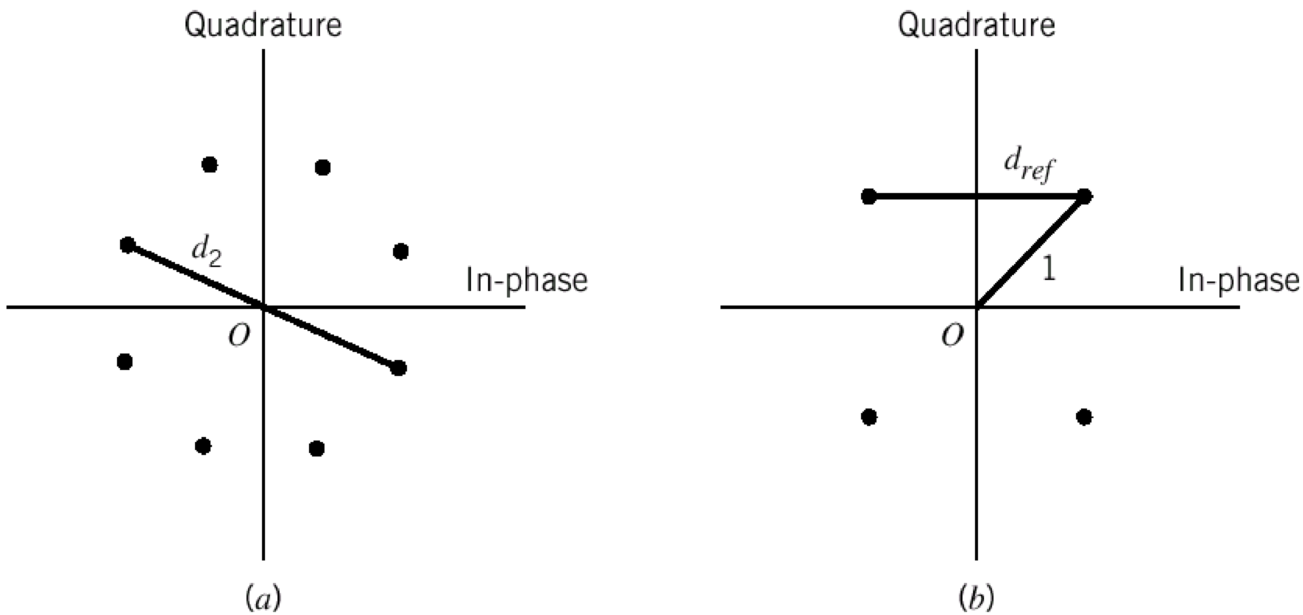


Figure 10.24
Single-space diagrams for calculation of asymptotic coding gain of Ungerboeck 8-PSK code. (*a*) Definition of distance d_2. (*b*) Definition of reference distance d_{ref}.

10.8 Turbo Codes

Good codes

algebraic structure → feasible decoding schemes

Problem of traditional codes (linear block codes & convolutional codes)

when approach the theoretical limit for Shannon’s channel capacity

code-word length (block codes)
constraint length (convolutional codes) ↑ →

computational complexity (ML decoder) ↑ exponentially → physically unrealizable

How to construct good codes with feasible decoding complexity?

10. 8 Turbo Codes

In the matter of channel coding and spectral efficiency, up to the invention of turbo codes, 3dB or more stood between what the theory promised and what real systems were able to offer. Ten years after the first publication on this new technique, turbo codes have commenced their practical service.

- C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-codes,” *Proc. ICC ’93*, Geneva, Switzerland, May 1993, pp. 1064–70.
- C. Berrou and A. Glavieux, “Near Optimum Error Correcting Coding and Decoding: Turbo-codes,” *IEEE Trans. Commun.*, vol. 44, no. 10, Oct. 1996, pp. 1261–71.
- C. Berrou, E. Bretagne, “The Ten-Year-Old Turbo Codes are Entering into Service,” *IEEE Communications Magazine,* Aug. 2003, pp. 110-116.

10.8.1 Turbo Coding

- Turbo encoder

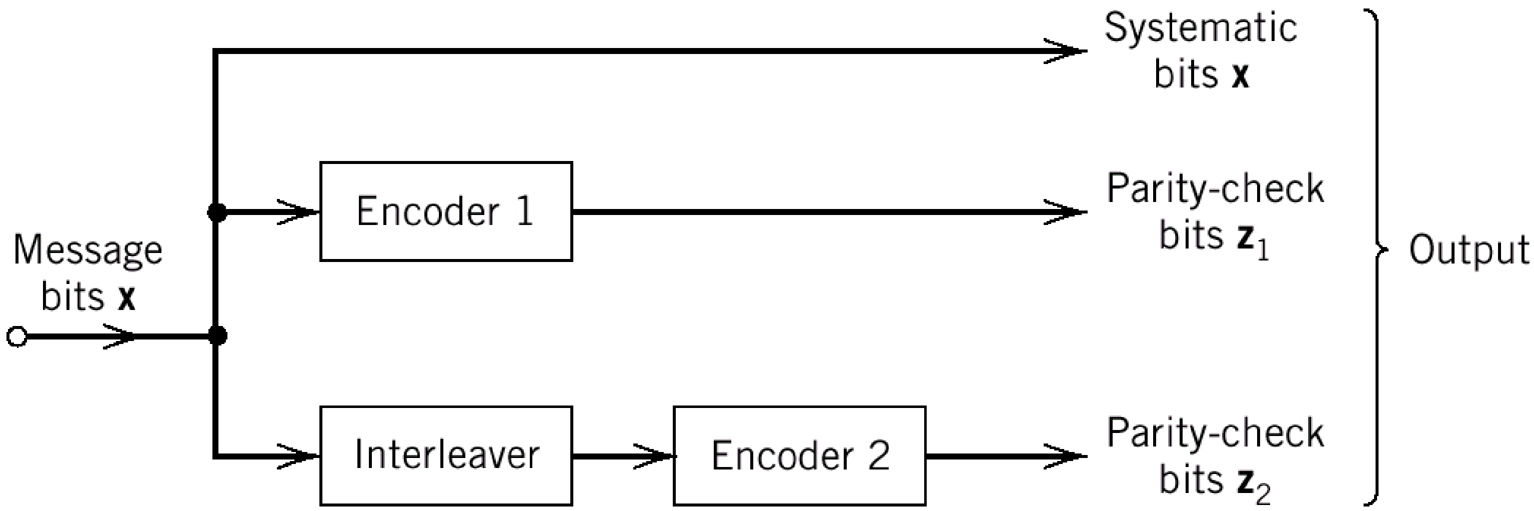


Figure 10.25

Block diagram of turbo encoder.

Back to 10.11

Back to 10.9

10.8.1 Turbo Coding

- Interleaver

output = input in a different temporal order

Types:

periodic, pseudo-random(used in turbo codes), ...

Reasons for the use of an interleaver:

1. Tie errors in one half(easily made) to errors in the other half(unlikely to occur)

2. Provide robust performance w.r.t mismatched decoding

10.8.1 Turbo Coding

- Encoder

Encoder 1 & 2 are the same (typically, but not necessarily)

short constraint-length *recursive systematic convolutional* (RSC) codes

Reasons for RSC:

recursive → internal state of the shift register depend on past outputs → affects the behavior of the error patterns → better performance

10.8.1 Turbo Coding

- Example 10.8 Eight-state RSC encoder

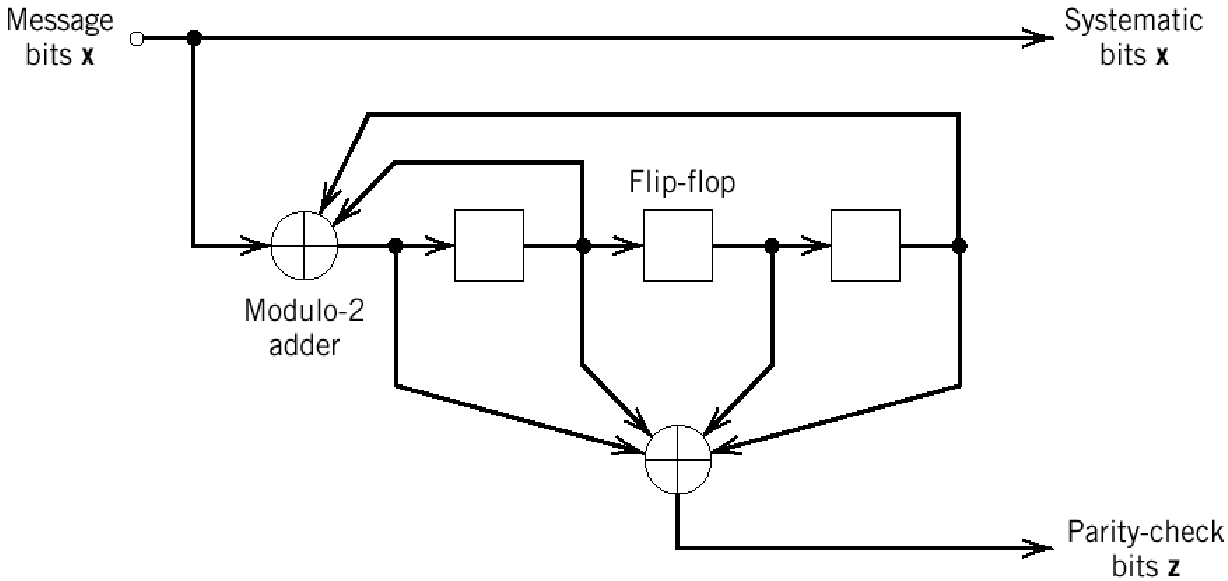


Figure 10.26

Example eight-state recursive systematic convolutional (RSC) encoder.

10.8.1 Turbo Coding

- Generator matrix

$$g(D) = \left[\ 1, \frac{1+D+D^2+D^3}{1+D+D^3}\ \right] \qquad (10.69)$$

The 2nd entry of g(D) is the transfer function of the feedback shift register.

In the time domain

$$m_i + m_{i-1} + m_{i-2} + m_{i-3} + b_i + b_{i-1} + b_{i-3} = 0 \qquad (10.70)$$

Equation (10.70) is the parity-check equation.

where $\{m_i\}$ denote the message sequence,

$\{b_i\}$ denote the parity sequence,

the addition is modulo-2.

10.8.1 Turbo Coding

- Turbo code -- linear block codes (block size determined by the size of the interleaver)

↓

How do we know the beginning & end of a code word ?

↓

1. Initialize to all-zero state before encoding.
2. *Add tail bits to return to all-zero sate.*

↓

1. Simple one -- just terminate the 1st RSC code leveling off in performance -- ‘error floor’ at low SNR
2. Refined one -- terminate both RSC codes reduce the ‘error floor’

10.8.1 Turbo Coding

- Punctured code

delete certain party check bits → increase the data rate

- Turbo encoder in figure 10.25

parallel encoding scheme

use of RSC code and pseudo-random interleaver

↓

Turbo codes:

1. Appear random to the channel → Shannon’s channel capacity

2. Possess sufficient structure for decoding to be physically realizable

10.8.2 Performance of Turbo Codes

- Simulation result shown in figure 10.27
- Parameters

channel : AWGN

code rate = 1/2

interleaver size = 65536

decoder -- BCJR algorithm

iteration numbers = 18

10.8.2 Performance of Turbo Codes

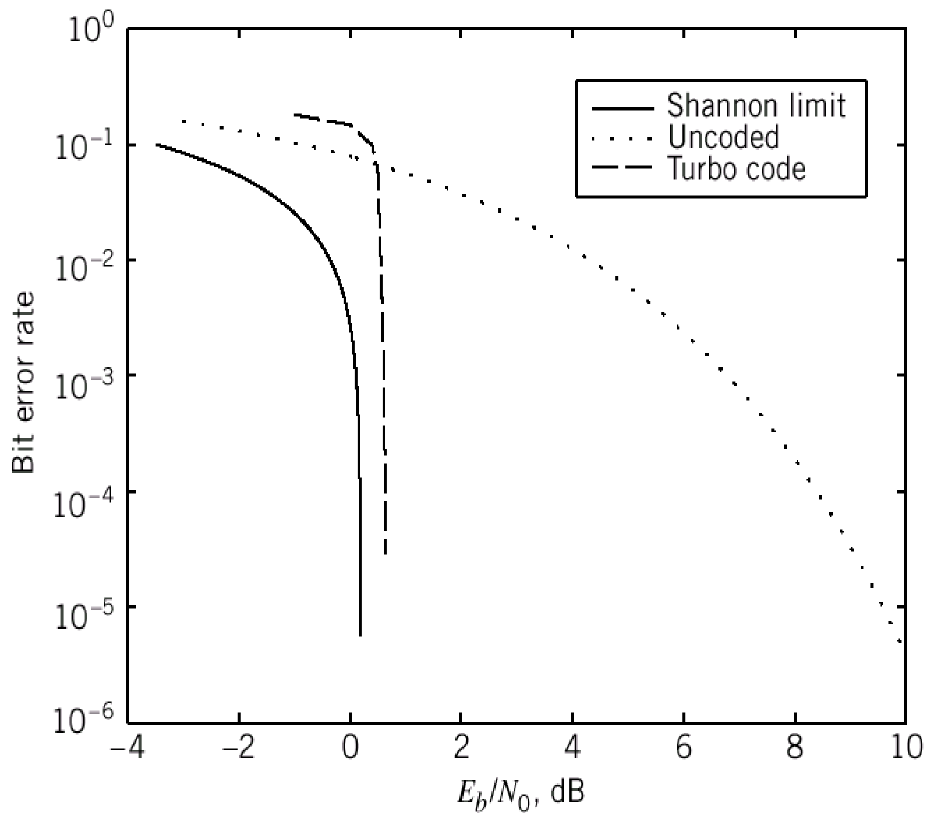


Figure 10.27

Noise performances of 1/2 rate, turbo code and uncoded transmission for AWGN channel; the figure also includes Shannon's theoretical limit on channel capacity for code rate $r = 1/2$.

10.8.2 Performance of Turbo Codes

- Conclusions :

1. BER (for the turbo-coded) > BER (for the uncoded) at low E_b/N_0

   BER for the turbo-coded drops very rapidly after reaching a critical value of E_b/N_0

2. At BER = 10^{-5} , turbo code from Shannon's theoretical limit < 0.5dB

3. Large size of interleaver(or block length of the code) + large number of iterations

impressive performance　　decoder complexity & latency

10.8.3 Turbo Decoding

- Decoding algorithm → 'turbo'
- Figure 10.28 structure of turbo decoder

two decoding stage (both use BCJR algorithm)

noisy systematic bits + noisy parity check bits

↓

an estimate of the original bits

- Differences between BCJR & Viterbi algorithm

1. BCJR SISO forward & backward recursion → complex

   Viterbi SIHO forward recursion

2. BCJR MAP decoder each bit → better performance

   Viterbi ML decoder whole sequence

Back

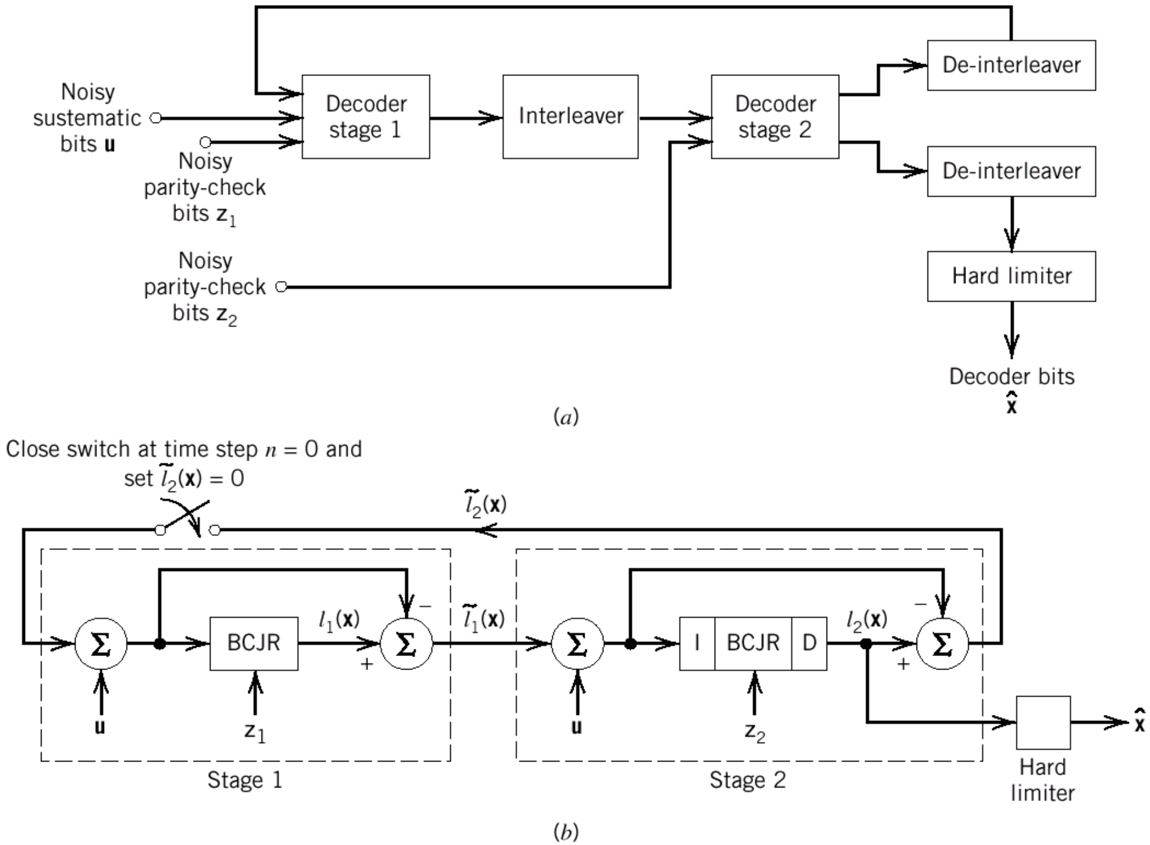


Figure 10.28 (*a*) Block diagram of turbo decoder. (*b*) Extrinsic form of turbo decoder, where I stands for interleaver, D for de-interleaver, and BCJR for BCJR algorithm for log-MAP decoding.

10.8.3 Turbo Decoding

- Assumptions in BCJR algorithm

  1. The channel encoding is modeled as a Markov process.

  2. The channel is memoryless.

- Notion of extrinsic information

  difference between two log-likelihood ratios (Figure 10.29)

  incremental information gained by a decoding stage

- Notion of intrinsic information

  a log-likelihood ratio fed back to input of the decoding stage

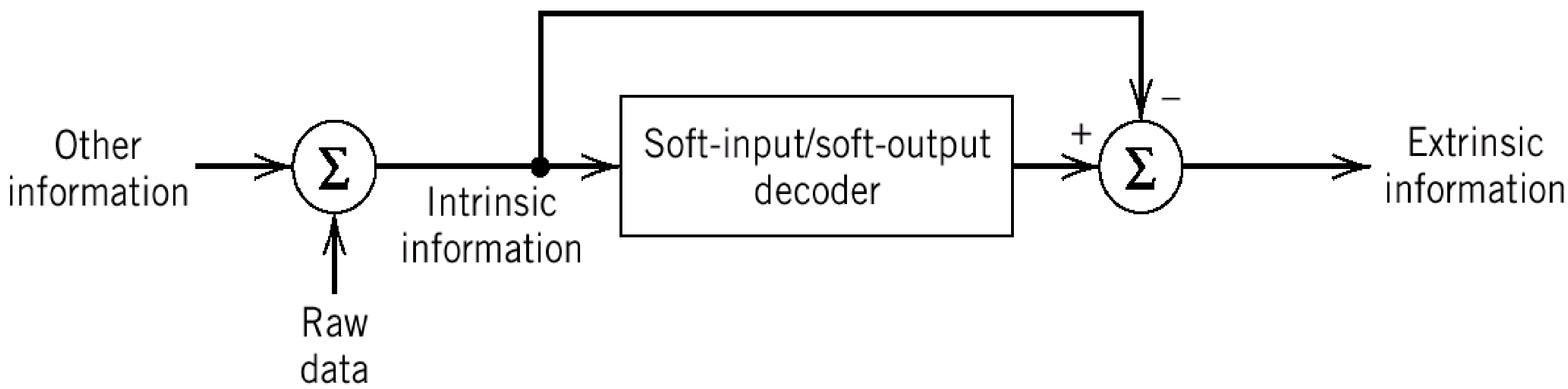


Figure 10.29

Illustrating the concept of extrinsic information.

10.8.3 Turbo Decoding

- 1st decoding stage

soft estimate of systematic bit x_j

$$l_1(x_j) = \log_2\left(\frac{P(x_j = 1 \mid \mathbf{u}, \xi_1, \tilde{l}_2(\mathbf{x}))}{P(x_j = 0 \mid \mathbf{u}, \xi_1, \tilde{l}_2(\mathbf{x}))}\right), \quad j = 1, 2, \ldots, k \qquad (10.71)$$

where u -- set of noisy systematic bits

ξ_1 -- set of noisy parity-check bits (by encoder 1)

$\tilde{l}_2(\mathbf{x})$-- extrinsic information from the 2nd stage

statistically independent

$$\rightarrow \quad l_1(\mathbf{x}) = \sum_{j=1}^{k} l_1(x_j) \qquad (10.72)$$

extrinsic information from the 1st stage

$$\tilde{l}_1(\mathbf{x}) = l_1(\mathbf{x}) - \tilde{l}_2(\mathbf{x}) \qquad (10.73)$$

10.8.3 Turbo Decoding

- 2nd decoding stage

soft estimate of systematic bit x_j

$$l_2(x_j) = \log_2\left(\frac{P(x_j = 1 \mid \mathbf{u}, \xi_2, \tilde{l}_1(\mathbf{x}))}{P(x_j = 0 \mid \mathbf{u}, \xi_2, \tilde{l}_1(\mathbf{x}))}\right), \qquad j = 1, 2, \ldots, k \tag{10.75}$$

where ξ_2 -- set of noisy parity-check bits (by encoder 2)

$\tilde{l}_1(\mathbf{x})$ -- extrinsic information from the 1st stage (reordered)

statistically independent

$\rightarrow$

$$l_2(\mathbf{x}) = \sum_{j=1}^{k} l_2(x_j)$$

extrinsic information from the 2nd stage

$$\tilde{l}_2(\mathbf{x}) = l_2(\mathbf{x}) - \tilde{l}_1(\mathbf{x}) \tag{10.74}$$

10.8.3 Turbo Decoding

- Estimate of the message bits x

$$\hat{\mathbf{x}} = \operatorname{sgn}(l_2(\mathbf{x})) \tag{10.76}$$

Note:

1. Initialization $\tilde{l}_2(\mathbf{x}) = 0$
2. Why feed only extrinsic information from one stage
   to the next?

   Maintain as much statistical independence between the bits as possible from one iteration to the next. If it is strictly true, the estimate $\hat{\mathbf{x}}$ approaches the MAP solution as the number of iterations → ∞.

10.8.4 The BCJR Algorithm

- BCJR algorithm -- MAP estimation

$x(t)$ -- input to a trellis encoder at time t

$y(t)$ -- corresponding output at the receiver

↓ (vector)

$\mathbf{y}_{(1,t)} = [y(1), y(2), \ldots, y(t)]$

$\lambda_m(t)$-- the probability that a a state $\mathbf{s(t)}$ of the

↓ trellis encoder equals m, where $m = 1 \sim M$

↓ (M-by-1 vector)

$$\boldsymbol{\lambda}(t) = \mathrm{P}[\mathbf{s}(\mathrm{t})|\mathbf{y}] \qquad (10.77)$$

10.8.4 The BCJR Algorithm

$$p(x(t)=1\,|\,\mathbf{y})=\sum_{s\in F_A}\boldsymbol{\lambda}_s(t) \qquad (10.78)$$

F_A -- the set of transitions that correspond to a symbol ‘1’ at the input

$\boldsymbol{\lambda}_s(t)$ -- *the s-component of* $\boldsymbol{\lambda}(t)$

forward estimation of state probabilities

$$\boldsymbol{\alpha}(t)=P(\mathbf{s}(t)\,|\,\mathbf{y}_{(1,t)}) \qquad (10.79)$$

backward estimation of state probabilities

$$\boldsymbol{\beta}(t)=P(\mathbf{s}(t)\,|\,\mathbf{y}_{(t,k)}) \qquad (10.80)$$

where $\mathbf{y}_{(t,k)} = [y(t), y(t+1), \ldots, y(k)]$

10.8.4 The BCJR Algorithm

- Separability theorem

$$\boldsymbol{\lambda}(t) = \frac{\boldsymbol{\alpha}(t) \bullet \boldsymbol{\beta}(t)}{\left\| \boldsymbol{\alpha}(t) \bullet \boldsymbol{\beta}(t) \right\|_1} \tag{10.81}$$

where

$$\boldsymbol{\alpha}(t) \bullet \boldsymbol{\beta}(t) = \begin{bmatrix} \alpha_1(t)\beta_1(t) \\ \alpha_2(t)\beta_2(t) \\ \vdots \\ \alpha_M(t)\beta_M(t) \end{bmatrix} \tag{10.82}$$

and the L_1 norm of $\boldsymbol{\alpha}(t) \bullet \boldsymbol{\beta}(t)$

$$\left\| \boldsymbol{\alpha}(t) \bullet \boldsymbol{\beta}(t) \right\|_1 = \sum_{m=1}^{M} \alpha_m(t)\beta_m(t) \tag{10.83}$$

The separability theorem says that the state distribution at time t given the past is independent of the state distribution at time t given the future. (Markovian assumption for channel encoding)

10.8.4 The BCJR Algorithm

state transition probability at time t

$$\gamma_{m',m}(t) = P[s(t) = m,\ \mathbf{y}(t) \mid s(t-1) = m'] \qquad (10.84)$$

M-by-M matrix of transition probabilities

$$\mathbf{\Gamma}(t) = \left\{\gamma_{m',m}(t)\right\} \qquad (10.85)$$

- recursion theorem

$$\boldsymbol{\alpha}^T(t) = \frac{\boldsymbol{\alpha}^T(t-1)\mathbf{\Gamma}(t)}{\left\|\boldsymbol{\alpha}^T(t-1)\mathbf{\Gamma}(t)\right\|_1} \qquad (10.86)$$

$$\boldsymbol{\beta}(t) = \frac{\mathbf{\Gamma}(t+1)\boldsymbol{\beta}(t+1)}{\left\|\mathbf{\Gamma}(t+1)\boldsymbol{\beta}(t+1)\right\|_1} \qquad (10.87)$$

The separability and recursion theorems together define the BCJR algorithm for the computation of a posteriori probabilities of the states and transitions of a code trellis, given the observation vector.

10.9 Computer Experiment: Turbo Decoding

- Two properties of Turbo codes
  - Property 1

    The error performance of the turbo decoder **improves with the number of iterations** of the decoding algorithm. This is achieved by feeding **extrinsic information from the output of the 1st decoding stage to the input of the 2nd decoding stage in the forward path** and feeding **extrinsic information form the output of the 2nd stage to the input of the 1st stage in the backward path,** and then permitting the iterative decoding process to take its natural course in response to the received noisy message and parity bits.

10.9 Computer Experiment: Turbo Decoding

- Property 2

The turbo decoder is capable of approaching the Shannon theoretical limit of channel capacity in a computationally feasible manner; this property has been demonstrated experimentally but not yet proven theoretically.

Property 2 requires that the block length of the turbo code be large.

10.9 Computer Experiment：Turbo Decoding

- Objective of the computer experiment

  demonstrate property 1

- Parameters

channel: AWGN

Turbo encoder : Fig.10.25

Encoder 1 : convolutional encoder [1,1,1]

Encoder 2 : convolutional encoder [1,0,1]

Block (i.e.interleaver) length: 1200bits

Turbo decoder: Fig.10.28

The BCJR algorithm for log-MAP decoding

10.9 Computer Experiment: Turbo Decoding

- Results

  Figure 10.30

- Observations

1. For fixed E_b/N_0, number of iterations ↑, Pe ↓ (confirming Property 1)
2. Iterations > 8, no significant improvement in decoding performance
3. For fixed number of iterations, E_b/N_0 ↑, Pe ↓

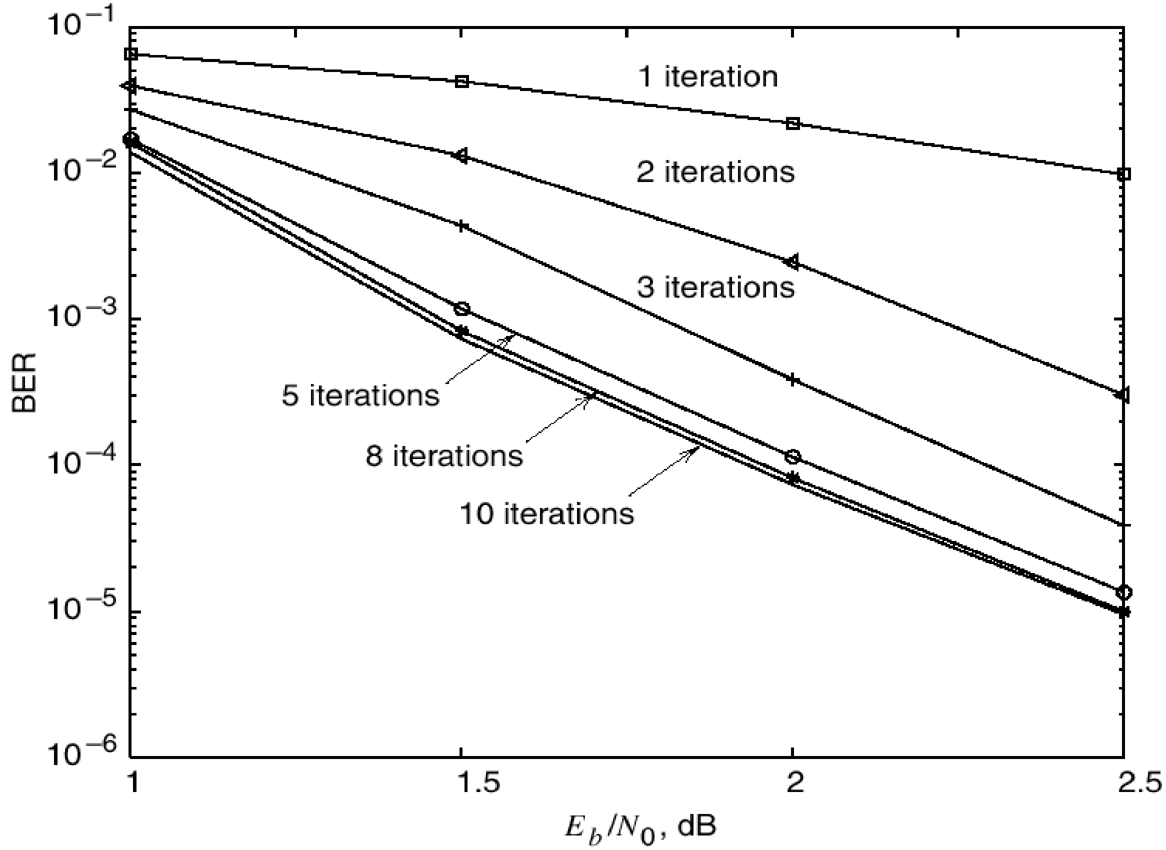


Figure 10.30

Results of the computer experiment on turbo decoding, for increasing number of iterations.

10.10 Low-Density Parity_Check Codes

- Turbo codes and LDPC codes belong to *compound codes*.
- Advantages of LDPC codes over Turbo codes

  1. Absence of low-weight code words.

     low-weight code words → error-floor problem

  2. Iterative decoding of lower complexity.

     Turbo codes: BCJR algorithm(computation scales linearly with the number of states, commonly >16)

     LDPC codes: parity-check trellis(2 states)

     parallelizable decoding

- Problem large block lengths →encoding complexity

10. 10. 1 Construction of LDPC Codes

Parity-check matrix *A* – sparse (mainly of 0s & a small number of 1s)

LDPC codes (n, t_c, t_r)

- n block length
- t_c weight in each column of *A*
- t_r weight in each row of *A*, and $t_r > t_c$

code rate $r = 1 - t_c/t_r$

Prove: Let ρ denote the density of 1s in *A*.

$$\rightarrow \quad t_c = \rho(n-k)$$

$$t_r = \rho n$$

$$\Rightarrow \quad \frac{t_c}{t_r} = 1 - \frac{k}{n}$$

10.10.1 Construction of LDPC Codes

The structure of LDPC codes is well portrayed by bipartite graphs (or Tanner graph).

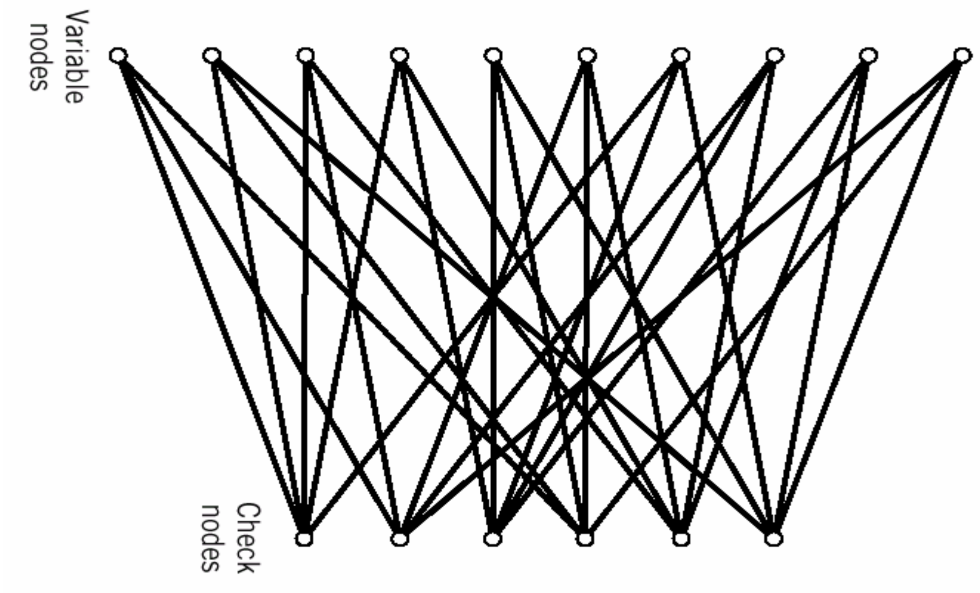


Figure 10.31
Bipartite graph of the (10, 3, 5) LDPC code.

10.10.1 Construction of LDPC Codes

Variable node	elements of the code word
Check node	parity-check equations
Regular	all variable(or check) nodes have the same degree (all the columns (or rows) of matrix A have the same number of 1s)
Low density	number of 1s << number of 0s in the parity-check matrix A
Degree of node	number of edges connected to another kind of node

degree of variable node = number of 1s in each column

degree of check node = number of 1s in each row

10.10.1 Construction of LDPC Codes

- The matrix A is constructed by putting 1s in A at random , subject to regularity constraints:

  1. Each column contains a small fixed number, t_c , of 1s.
  2. Each row contains a small fixed number, t_r , of 1s.

  In practice, these regularity constraints are often violated slightly in order to avoid having linearly dependent rows in the parity-check matrix A.

- LDPC code is not systematic.

10.10.1 Construction of LDPC Codes

- Gaussian elimination method for deriving a generator matrix G

1. partition

$$c = [\, b \quad m \,]$$

where c 1-by-n code vector

b 1-by-(n-k) parity vector

m 1-by-k message vector

$$\mathbf{A}^T = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix} \tag{10.89}$$

Where A_1 (n-k)-by-(n-k) square matrix

A_2 k-by-(n-k) rectangular matix

10.10.1 Construction of LDPC Codes

Imposing the constraint of linear block

$$\longrightarrow \quad [\mathbf{b} \quad \mathbf{m}]\begin{bmatrix}\mathbf{A}_1 \\ \mathbf{A}_2\end{bmatrix} = \mathbf{0}$$

$$\longrightarrow \quad \mathbf{b}\mathbf{A}_1 + \mathbf{m}\mathbf{A}_2 = \mathbf{0} \qquad (10.90)$$

$$\because \quad \mathbf{b} = \mathbf{m}\mathbf{P}$$

$$\longrightarrow \quad \mathbf{P}\mathbf{A}_1 + \mathbf{A}_2 = \mathbf{0}$$ (for nonzero vector m)

$$\therefore \quad \mathbf{P} = \mathbf{A}_2\mathbf{A}_1^{-1} \qquad (10.91)$$

$$\longrightarrow \quad \mathbf{G} = [\mathbf{P} \quad \mathbf{I}_k] = [\mathbf{A}_2\mathbf{A}_1^{-1} \quad \mathbf{I}_k] \qquad (10.92)$$

10.10.1 Construction of LDPC Codes

- **Note:**

  If we take the matrix A for some arbitrary LDPC code and just pick (n-k) columns of A at random to form a square matrix A_1, there is no guarantee that A_1 will be nonsingular (i.e., the inverse $\mathbf{A}_1^{-1}$ will exist), even if the rows of A are linearly independent.

10.10.1 Construction of LDPC Codes

➤ Example 10.9 (10,3,5) LDPC Code

Bipartite graph Figure 10.31

Parity-check matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{\mathbf{A}_1^T} \qquad \underbrace{\qquad\qquad\qquad}_{\mathbf{A}_2^T}$$

10.10.1 Construction of LDPC Codes

- Example 10.9 (10,3,5) LDPC Code

The inverse of matrix A_1

$$\mathbf{A}_1^{-1} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

10.10.1 Construction of LDPC Codes

- Example 10.9 (10,3,5) LDPC Code

The matrix product $\mathbf{A}_2\mathbf{A}_1^{-1}$

$$\mathbf{A}_2\mathbf{A}_1^{-1} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

10.10.1 Construction of LDPC Codes

➢ Example 10.9 (10,3,5) LDPC Code

The generator matrix G

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

$$\underbrace{\qquad\qquad\qquad\qquad}_{\mathbf{A}_2\mathbf{A}_1^{-1}} \qquad \underbrace{\qquad\qquad\qquad}_{\mathbf{I}_\mathrm{k}}$$

10.10.1 Construction of LDPC Codes

Note:

1. This example is intended only for the purpose of illustrating the procedure involved in the generation of a LDPC code. In practice, the block length n is orders of magnitude larger than that considered in this example.
2. Another constraint is used in constructing the matrix A to improve the performance of LDPC codes. That is, constrain all pairs of columns to have a matrix overlap (i.e., inner product of any two columns in matrix A) not to exceed 1.

10.10.2 Minimum Distance of LDPC Codes

block length of LDPC code $10^3 \sim 10^6$

↓

algebraic analysis of LDPC codes is difficult

↓

statistical analysis of LDPC codes

minimum distance of a member code – random variable

Note: It is shown that as the block length n increases, for fixed $t_c \geq 3$ and $t_r \geq t_c$, the probability distribution of the minimum distance can be overbounded by a function that approaches a unit step function at a fixed fraction $\Delta_{t_c t_r}$ of the block length n. Thus, for large n, practically all the LDPC codes in the ensemble have a minimum distance of at least $n\Delta_{t_c t_r}$. (*Table 10.10*)

10.10.3 Probabilistic Decoding of LDPC Codes

transmitted vector $\mathrm{c} = \mathrm{mG}$

received vector $\mathrm{r} = \mathrm{c} + \mathrm{e}$

error vector e

bit-by-bit decoding: find the most probable vector $\hat{\mathbf{c}}$ that satisfies the condition $\hat{\mathbf{c}}\mathbf{A}^T = 0$.

set of bits that participate in check i $\phi(i)$

set of checks in which bit j participates $\varphi(j)$

A set $\phi(i)$ that excludes bit j $\phi(i) \setminus j$

A set $\varphi(j)$ that excludes check i $\varphi(j) \setminus i$

10.10.3 Probabilistic Decoding of LDPC Codes

The decoding algorithm has two alternating steps:

1. Horizontal step   run along the rows of *A*
2. Vertical step   run along the columns of *A*

Two probabilistic quantities associated with nonzero elements of *A* are alternately updated.

P_{ij}^{x} the probability that bit *j* is symbol x(0 or 1), given the information derived via checks performed in the horizontal step, except for check *i*

Q_{ij}^{x} the probability that check *i* is satisfied, given that bit *j* is fixed at the value x and the other bits have the probabilities $P_{ij'} : j' \in \phi(i) \setminus j$

10.10.3 Probabilistic Decoding of LDPC Codes

- Sum-product algorithm
  - Initialization

Set $P_{ij}^0 = p_j^0, P_{ij}^1 = p_j^1$ with $p_j^0 + p_j^1 = 1$

  - Horizontal step

Define $\Delta P_{ij} = P_{ij}^0 - P_{ij}^1$

For each weight-pair (i, j), compute

$$\Delta Q_{ij} = \prod_{j' \in \phi(i) \setminus j} \Delta P_{ij'}$$

Set

$$Q_{ij}^0 = \frac{1}{2}(1 + \Delta Q_{ij})$$

$$Q_{ij}^1 = \frac{1}{2}(1 - \Delta Q_{ij})$$

10.10.3 Probabilistic Decoding of LDPC Codes

– Vertical step

For each bit *j*, compute

$$P_{ij}^0 = \alpha_{ij} p_j^0 \prod_{i' \in \varphi(j) \setminus i} Q_{i'j}^0$$

$$P_{ij}^1 = \alpha_{ij} p_j^1 \prod_{i' \in \varphi(j) \setminus i} Q_{i'j}^1$$

Where the scaling factor α_{ij} is chosen to make

$$P_{ij}^0 + P_{ij}^1 = 1$$

The pseudo-posterior probabilities are updated

$$P_j^0 = \alpha_j p_j^0 \prod_{i \in \varphi(j)} Q_{ij}^0$$

$$P_j^1 = \alpha_j p_j^1 \prod_{i \in \varphi(j)} Q_{ij}^1$$

Where α_j is chosen to make

$$P_j^0 + P_j^1 = 1$$

10.10.3 Probabilistic Decoding of LDPC Codes

The quantities obtained in the vertical step are used to compute a tentative estimate $\hat{\mathbf{c}}$.

If the condition $\hat{\mathbf{c}}\mathbf{A}^T = 0$ is satisfied, the decoding algorithm is terminated.

Otherwise, the algorithm goes back to the horizontal step. If after some maximum number of iterations (e.g., 100 or 200) there is no valid decoding , a decoding failure is declared.

The sum-product algorithm passed probabilistic quantities between the check nodes and variable nodes of the bipartite graph.

10.10.3 Probabilistic Decoding of LDPC Codes

Complexity:

LDPC decoders are simpler to implement than turbo decoders, since each parity-check constraint can be represented by a simple convolutional coder with one bit of memory.

Performance:

In light of experimental results reported in the literature: regular LDPC codes do not appear to come as close to Shannon’s limit as do their turbo code counterparts.

10.11 Irregular Codes

- ✓ Regular codes

  Turbo codes in section 10.8

  LDPC codes in section 10.10

- ✓ Irregular codes

  The error correcting performance can be improved substantially by using their irregular forms.

  Irregular turbo codes

  standard turbo code encoder (Figure 10.25)

  irregular turbo code encoder (Figure 10.32)

10.11 Irregular Codes

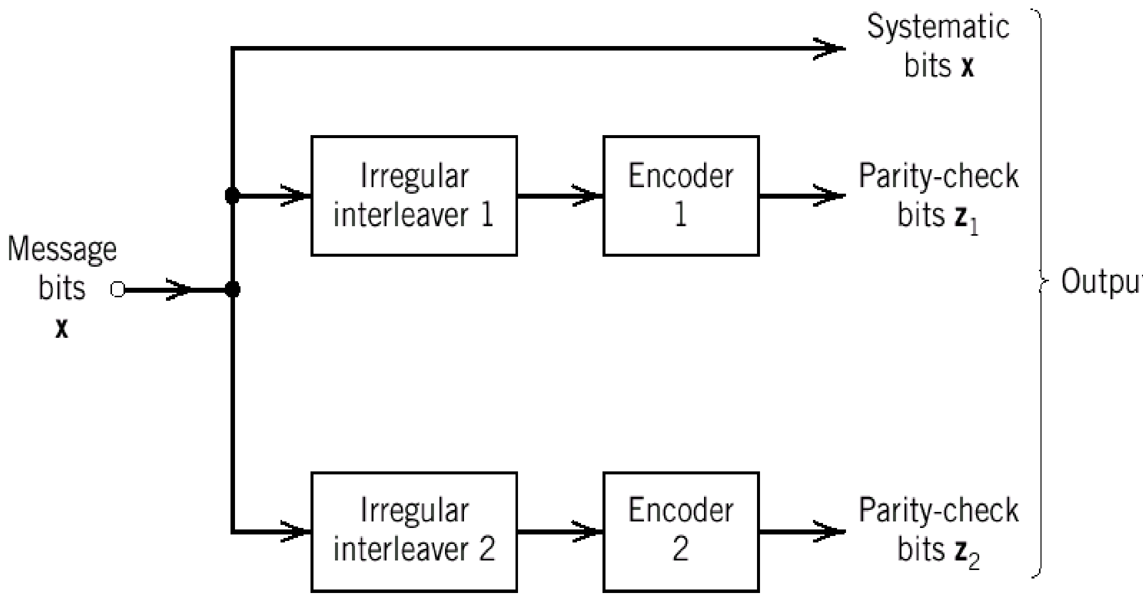


Figure 10.32
Block diagram of irregular turbo encoder.

10.11 Irregular Codes

- Regular interleaver

  mapping each systematic bit to a unique input bit of the convolutional encoder

- Irregular interleaver

  mapping some systematic bits to multiple input bits of the convolutional encoder

- Decoding

  in a similar fashion to regular turbo codes

10.11 Irregular Codes

Irregular LDPC codes

The degrees of the variable and check nodes in the bipartite graph are chosen according to some distribution. (an example)

performance comparison: (Figure 10.33)

- Irregular LDPC code: k=50,000, n=100,000, rate=1/2
- Turbo code (regular): k=65,536, n=131,072, rate=1/2
- Irregular turbo code: k=65,536, n=131,072, rate=1/2

convolutional encoders in turbo codes:

Encoder 1: $g(D) = 1 + D^4$

Encoder 2: $g(D) = 1 + D + D^2 + D^3 + D^4$

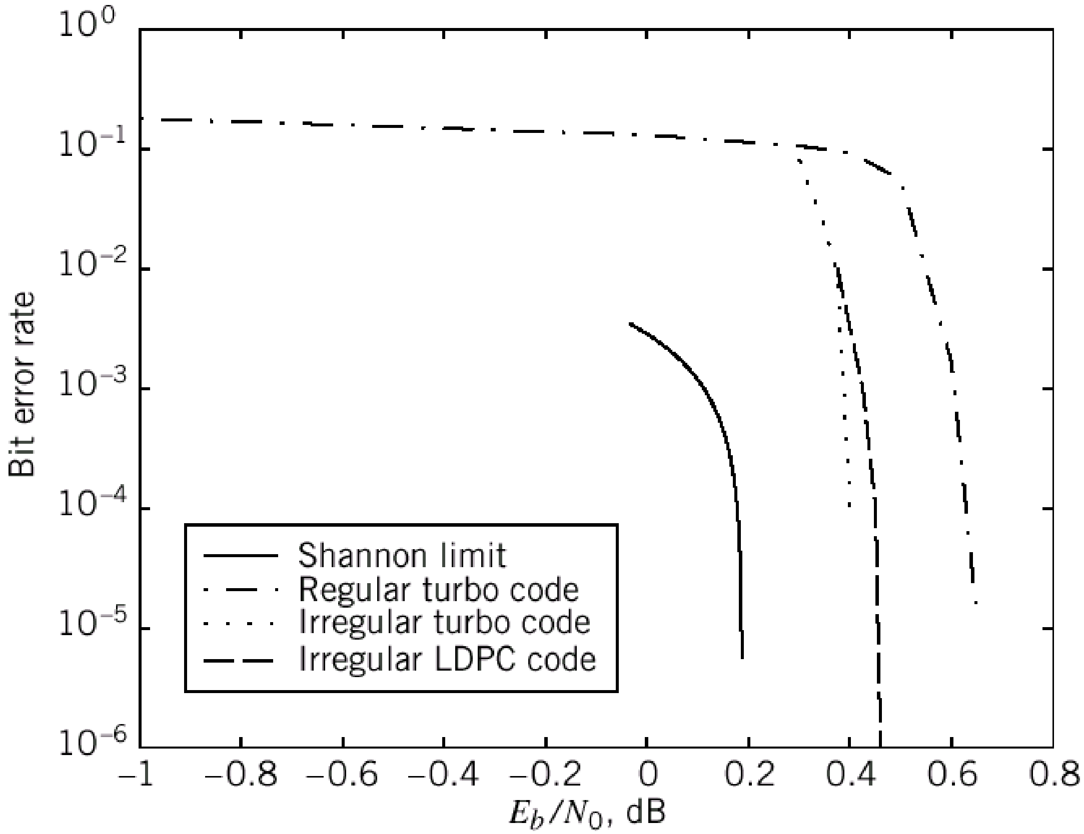


Figure 10.33

Noise performances of regular turbo code, irregular turbo code and irregular low-density parity-check (LDPC) code, compared to the Shannon limit for code rate $r = 1/2$.

10. 11 Irregular Codes

- Observations:

1. The irregular LDPC code outperforms the regular turbo code in that it comes closer to Shannon's theoretical limit by 0.175dB.
2. Among the three codes displayed therein, the irregular turbo code is the best in that it is just 0.213 dB away form Shannon's theoretical limit.

10.12 Summary and Discussion

Error-control coding techniques may be divided into two broadly defined families:

1. Algebraic codes

rely on abstract algebraic structure built into the design of the codes for decoding at the receiver.

include: Hamming codes, maximal-length codes, BCH codes, and Reed-Solomon codes

properties: linearity property
cyclic property

10.12 Summary and Discussion

2. Probabilistic codes

rely on probabilistic methods for their decoding at the receiver.

include: trellis codes, turbo codes, and low-density parity-check codes

two basic methods the decoding based on:

1. Soft input hard output
   used by Viterbi algorithm
   MLS estimation　　　trellis codes
2. Soft input soft output
   used by BCJR algorithm
   MAP estimation　turbo codes & LDPC codes

10.12 Summary and Discussion

- TCM

  Convolutional encoding and modulation are combined.

  significant coding gains over uncoded multilevel modulation without sacrificing bandwidth efficiency

- Properties of turbo codes & LDPC codes

  1. Random encoding of a linear block kind.

  2. Error performance within a hair's breadth of Shannon's theoretical limit on channel capacity in a physically realizable fashion.

- Coding gains → bit rates ↑

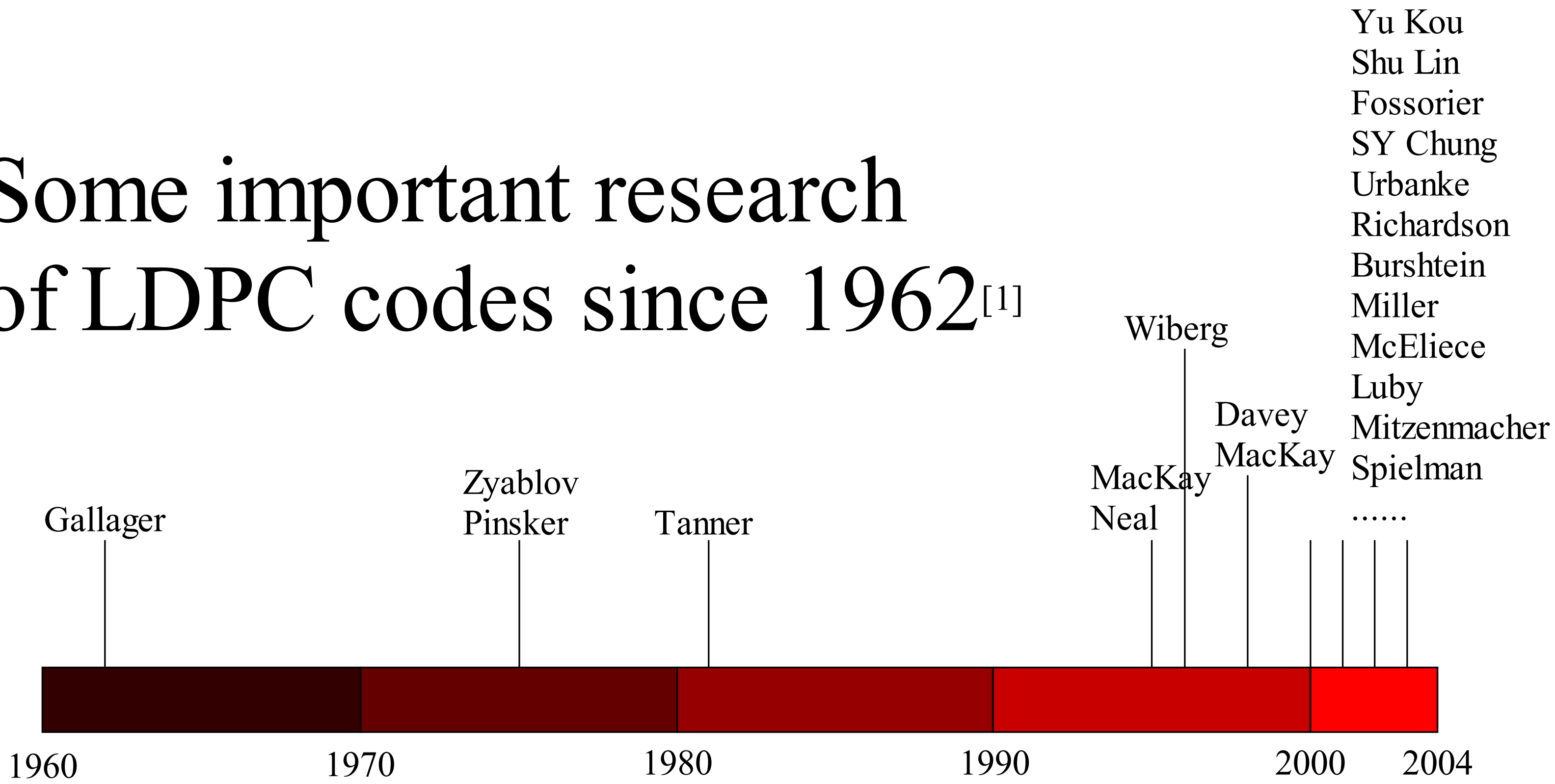
  (e.g. 10dB) or transmitted signal energy per symbol ↓

  or probability of error ↓

一、LDPC码的发展历史

- **1962**年前后**Gallager**首先提出**LDPC**码**[1,2]**，并给出**LDPC**码的简单构造和硬判决概率译码；
- **1981**年**Tanner**建立了编码的图模型概念**[6]**，证明了和积算法在无环图中译码的最佳性并提出了构造适合和积译码的图模型的代数方法；
- **1996**年前后**MacKay**和**Neal[12][3]**，**Spiser**和**Spielman[10]**，**Wiberg[11]**重新发现了**LDPC**码的良好性能；
- **1998**年**Davey**和**MacKay**提出了基于**GF(q)**的**LDPC**码**[4]**；
- **1998**年**Luby**等人提出了基于非正则图的**LDPC**码**[5];**
- **2001**年专辑“基于图的码和迭代解码”**,IEEE Trans. on Inform. Theory,vol.47,Feb.2001;**
- **2004**年文集“**LDPC**码构造等”， **IEEE Trans. on Inform. Theory, vol.50,**June 2004.

Some important research of LDPC codes since 1962[1]



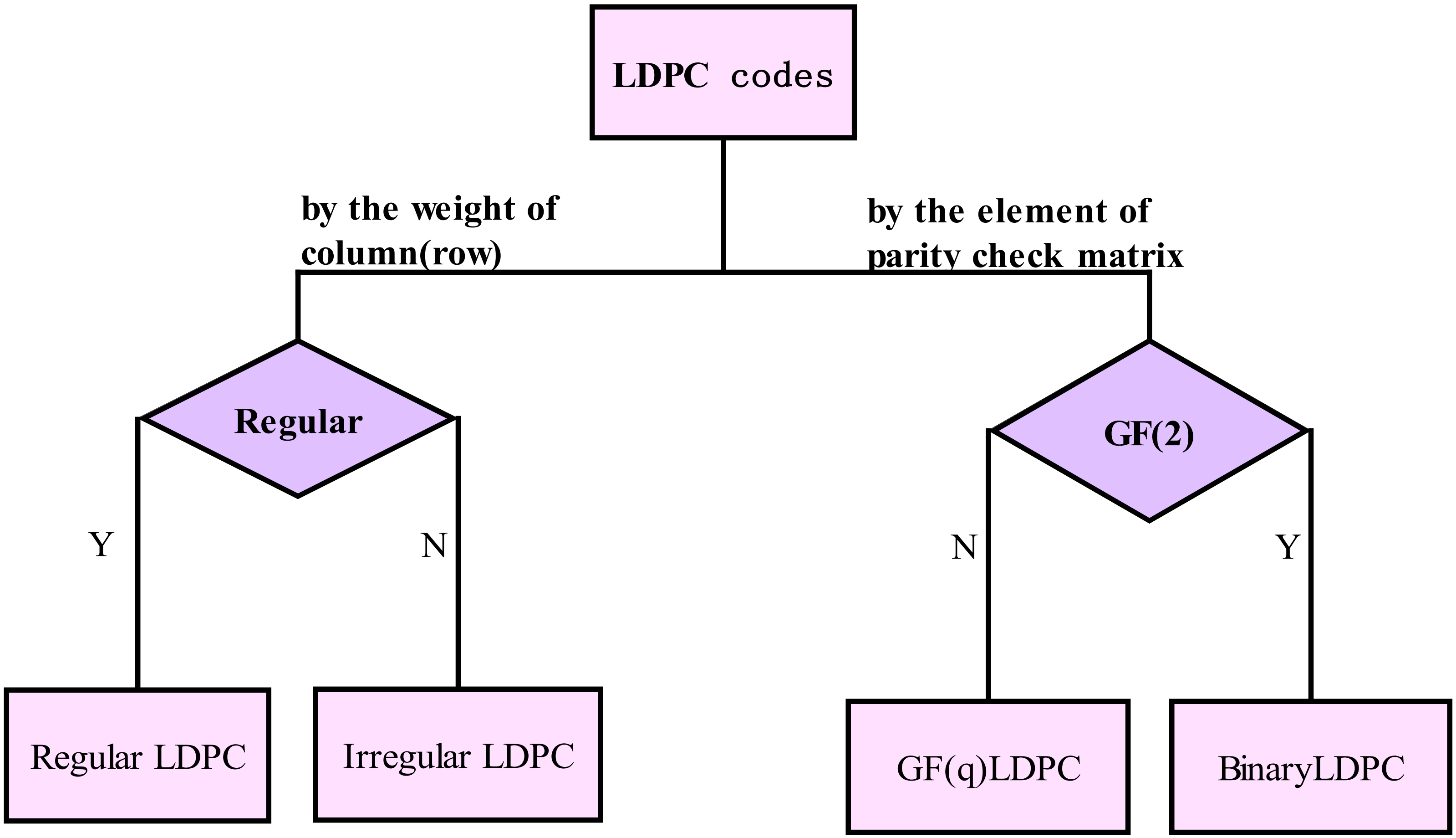
二、LDPC码基础简介

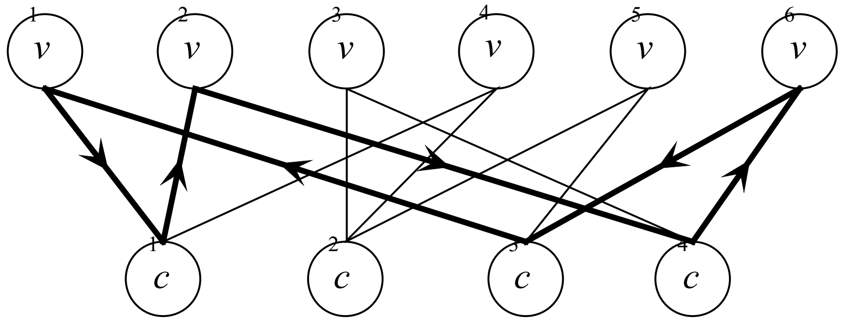
LDPC码就是一种普通的线性分组码，可以用生成矩阵和校验矩阵来表征；

LDPC码又是一种特殊的分组码，特殊性就在于它的奇偶校验矩阵中‘1’的数目远小于‘0’的数目，称为稀疏性，“低密度”也来源于此；

LDPC码又称为稀疏图码，它可以用一个二分图来表征，在图论中一个图是由顶点和边组成的，二分图：图中所有的顶点分为两个子集，任何一个子集内部各个顶点之间没有边相连，任意一个顶点都和一个不在同一个子集里的顶点相连；

LDPC码的二分图又称为Tanner图，这是由Tanner在1982年首次用它来表示低密度码的，一个Tanner图和一个校验矩阵完全对应；


LDPC codes
by the weight of
column(row)
by the element of
parity check matrix
Regular
GF(2)
Y
N
N
Y
Regular LDPC
Irregular LDPC
GF(q)LDPC
BinaryLDPC



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

这是一个简单的(6,3,3)码字的二分图，上面是变量节点，下面为校验节点；

圈：由变量节点、校验节点和边首尾相连组成的闭合环路，即文献中的cycle；

Girth的定义：码字二分图或Tanner图中最短圈的圈长；

图中粗线构成了一个长度为6的圈，如果没有长度为4的圈，那么这个图的最小圈长，即girth，就是6；

给出一个码的校验矩阵后，可以用文献[19]中Mao Yongyi提的搜索方法得到该LDPC码的girth及其分布。

为什么提到LDPC码的girth？

答：因为目前LDPC码的最好的解码方法是Sum-Product算法或者Belief Propagation (BP)算法，这种算法是一种渐进最大似然的算法，如果Tanner图中不存在圈，也就是girth是无穷大，则这种算法等效于最大似然算法；当然，在码长固定的情况下，对于适用的码字来说无圈是不可能的；但是通过增大最小圈的圈长，也就是增大girth，可以提高码字的性能，girth达到一定的值就可以接近无圈时的性能。

显然决定码字性能的是码间距，但是我们无法去直接约束控制码间距；控制Tanner图的girth虽然也有很大的难度，但却是可以通过一些方法消除长度较短的圈；一般来说，girth大的码字其码间距也大，但是码间距大的其girth不一定就很大。

因此，girth是目前设计LDPC码最常用到的关键词之一。

LDPC 码所面临的一个主要问题是其较高的编码复杂度和编码时延。对其采用普通的编码方法，LDPC码具有二次方的编码复杂度，在码长较长时这是难以接受的，幸运的是校验矩阵稀疏性使得LDPC码的编码成为可能。

循环码或者准循环码的编码复杂度由于和码长成线性关系，它们的编码复杂度最低；

从性能上考虑，具有大的最小距离的码字有很多都落在准循环码这个集合里，因此用好的方法找出这些具有较大最小距离的准循环码是LDPC码研究的一个热点。

Decoding Algorithms

- Bit Flipping Algorithm （BF）
- Weighted Bit Flipping Algorithm （WBF）
- Belief Propagation Algorithm（BP）
- Min Sum Algorithm（MS）
- OSD+BP Algorithm

Belief Propagation algorithm

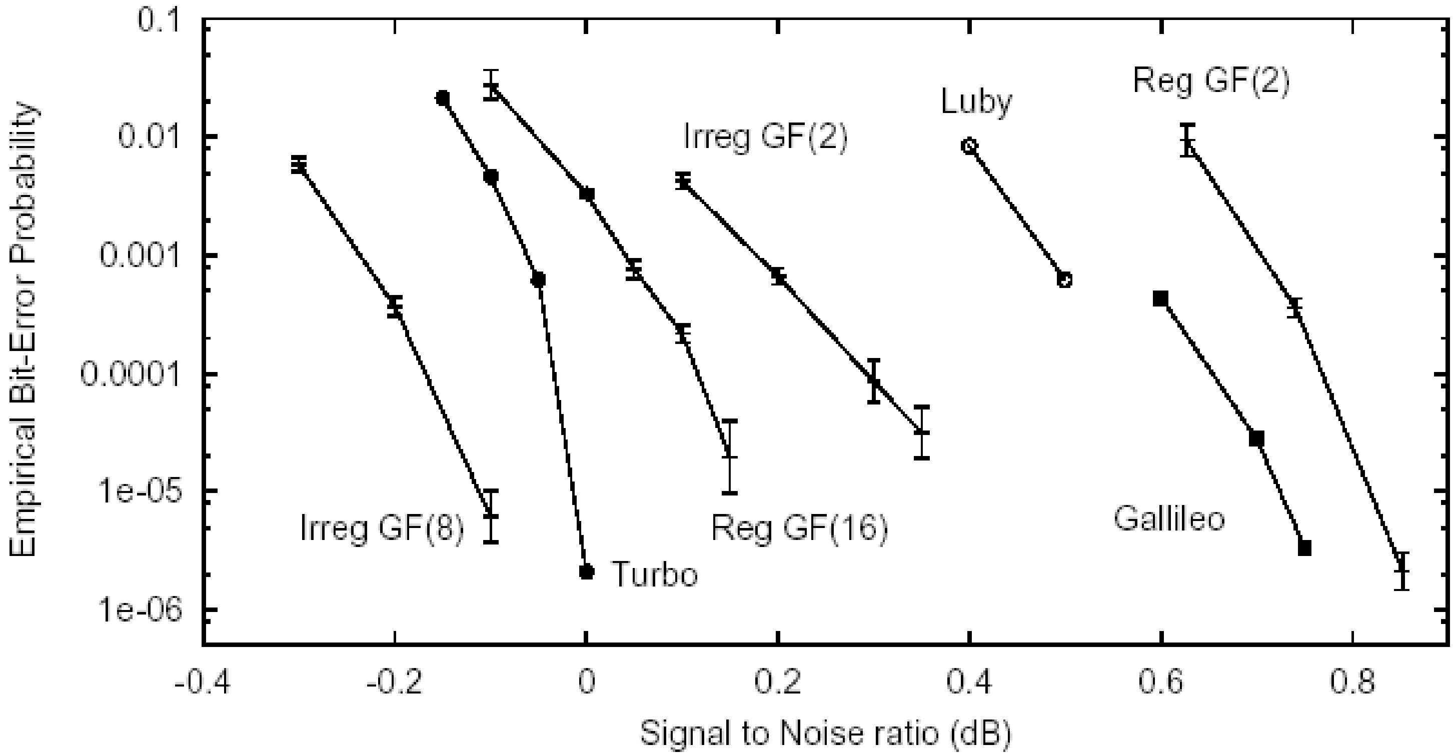
- All the effective decoding strategies for LDPC codes are message passing algorithms
- The best algorithm known is the Belief Propagation algorithm

(1) Complicated calculations are distributed among simple node processors

(2) After several iterations, the solution of the global problem is available

(3) BP algorithm is the optimal if there are no cycles or ignore cycles

LDPC codes performance



———rate=1/4, AWGN Channel, Thesis of M.C.Davey