UESTC4004
Digital Communications

Instructors: Prof. Muhammad Imran and Sajjad Hussain

This week: Sajjad Hussain sajjad.hussain@glasgow.ac.uk

Mid Exam

- One hour exam on ???
- Course includes Week 1 and Week 2 lectures
- It carries **10%** weight
- Sample available on Moodle

Lab updates

- This week you'll get your Lab 2 checked & work on Lab 3 and get Lab 3 checked, if possible.
- Lab 3 and Lab 4 (Project) are designed in a way to allow you to work from **home**, therefore it is expected that you'll complete them before coming to Lab 4 (Week 15).
- In the Lab 4, you'll have the opportunity to get your Lab 3 checked by the GTAs while I'll evaluate your **project understanding** (2-3 minute viva) in parallel.
- Completed Lab document (Lab 1, 2 & 3) and Project report submission deadline is **20 December** through Moodle.

Lab 3 - Briefing

Section 11.4 – Carrier Synchronization

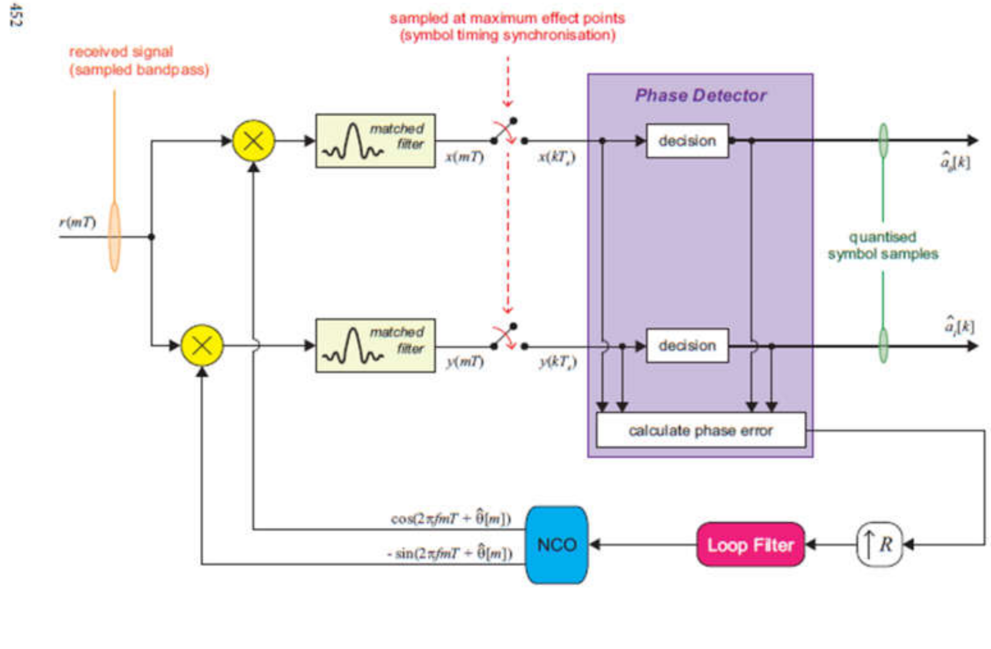


Figure 11.17: Carrier synchronisation (synchronising at demodulation stage) [38]

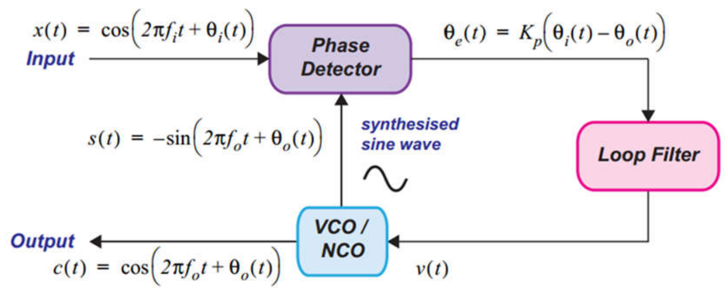


Figure 7.8: Block diagram of a PLL

Similar to what we studied in Week 2 – PLL

Lab 3 - Briefing

Section 11.8 – Course Carrier Synchronization

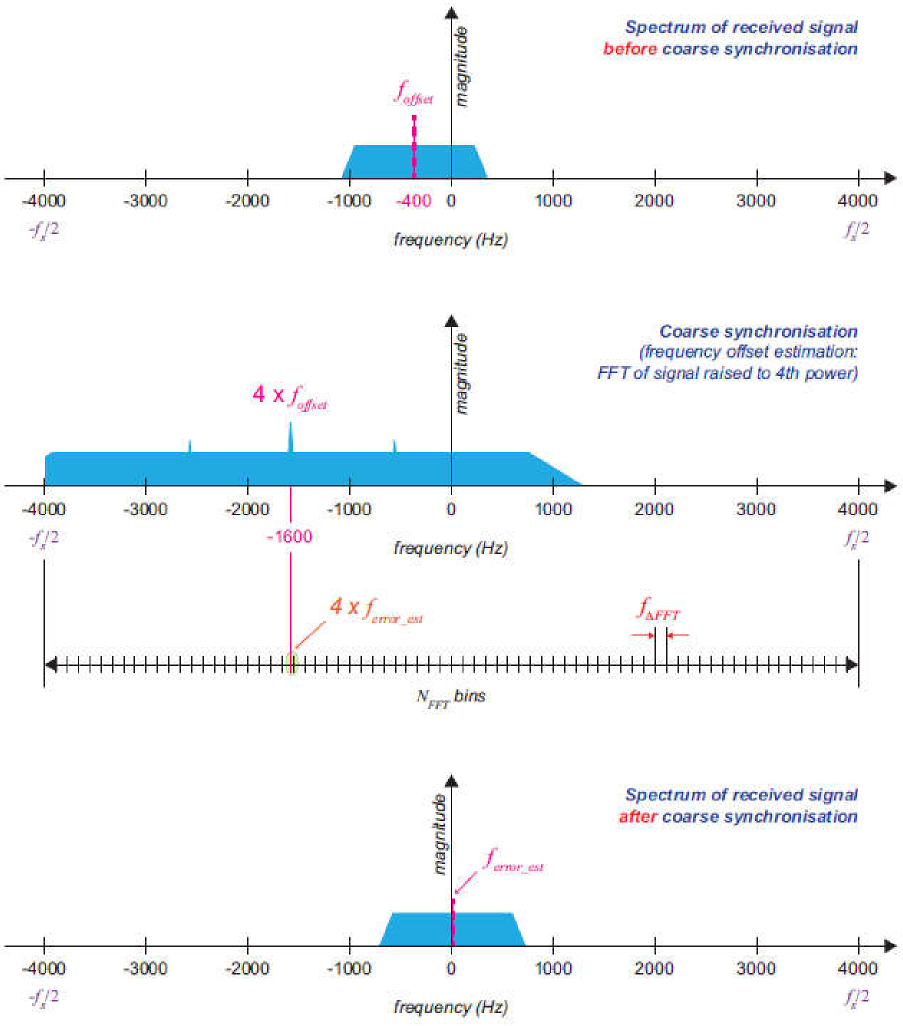


Figure 11.36: Coarse frequency correction using the FFT of the 4th power of a QPSK signal

Exploiting the signal structure properties – For QPSK, M=4

Channel Coding

- Adding redundant bits to improve immunity against channel noise and distortions. The purpose is to correct or at least detect the errors introduced due to channel in our information signal.
- Parity bit check is the simplest example of channel coding.

6.2 Types of Error control

- Error detection and retransmission
- Forward error correction

Terminal Connectivity

- Simplex
- Half duplex
- Full duplex

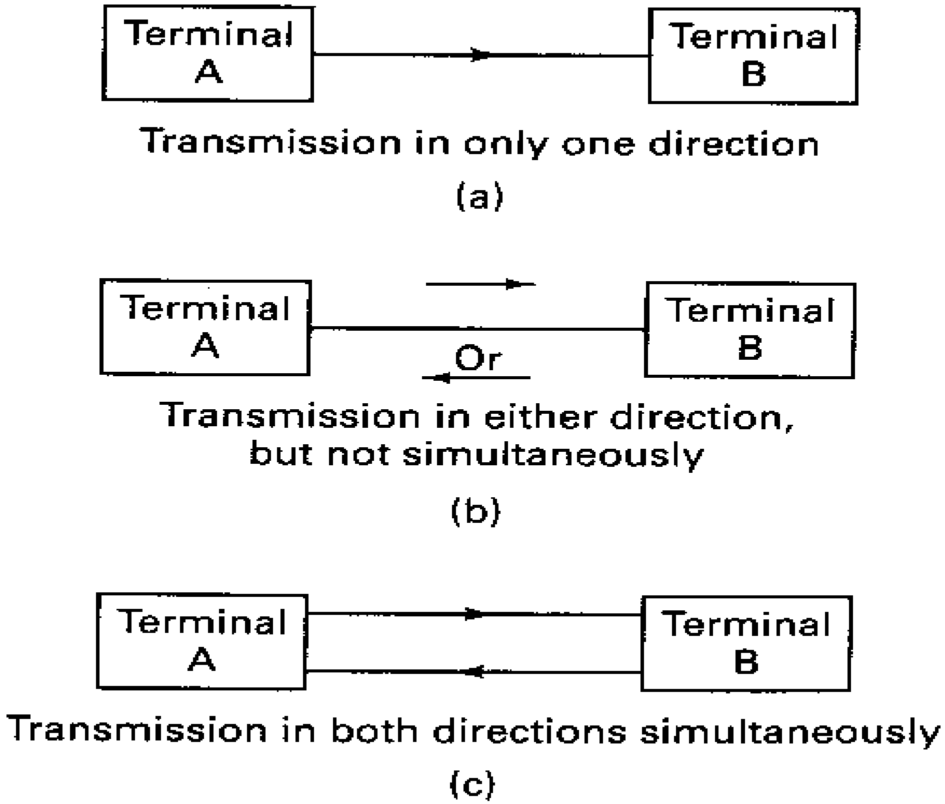


Figure 6.6: Terminal connectivity classifications (a) Simplex (b) Half-duplex (c) Full-duplex

Automatic Repeat Request

- ARQ vs. FEC
  - ARQ is much simpler than FEC and need no redundancy.
  - ARQ is sometimes not possible if
    - A reverse channel is not available or the delay with ARQ would be excessive
    - The retransmission strategy is not conveniently implemented
    - The expected number of errors, without corrections, would require excessive retransmissions

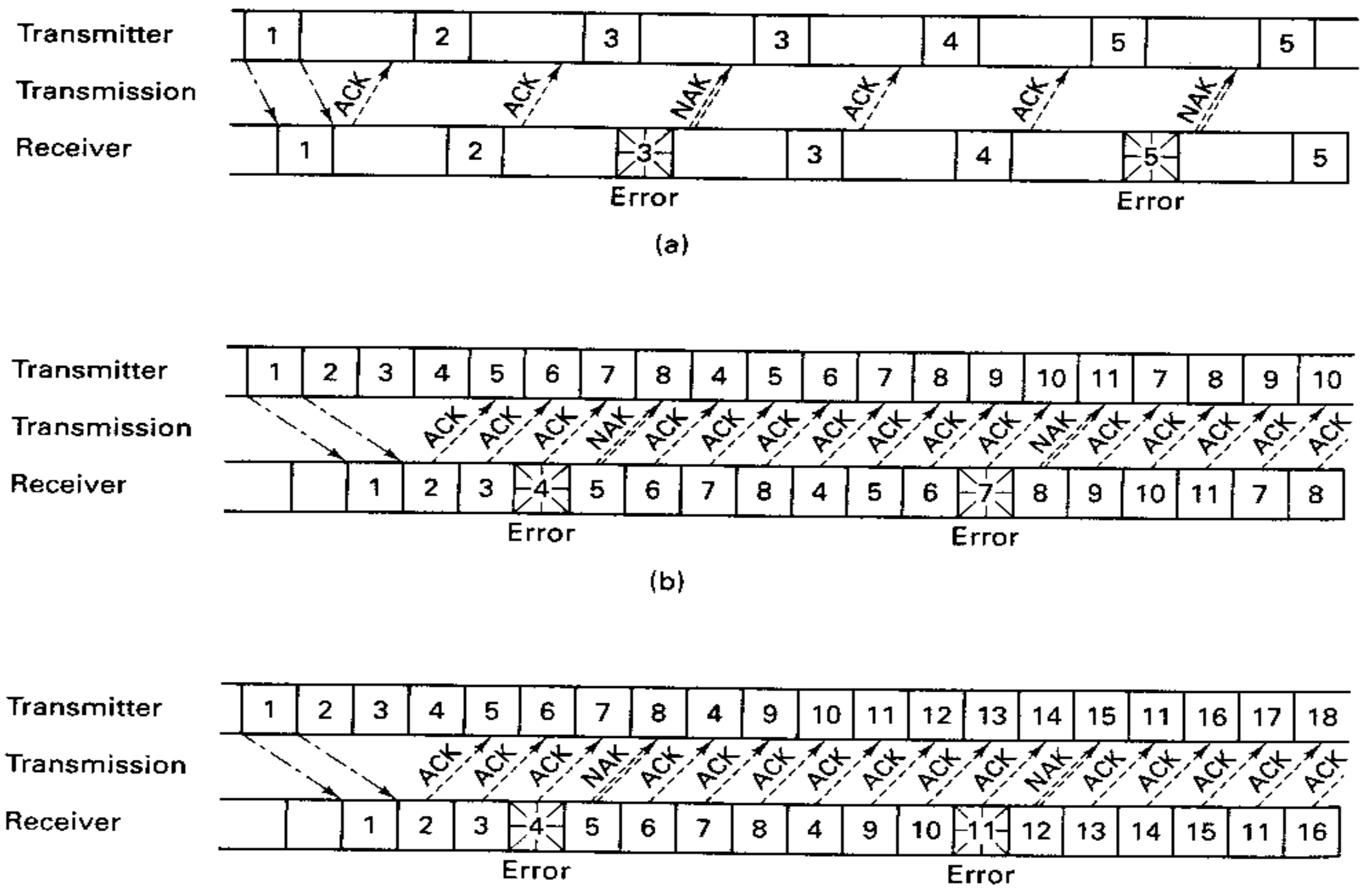


Figure 6.7: Automatic Repeat Request (ARQ) (a) Stop and wait ARQ (b) Continuous ARQ with pullback (c) Continuous ARQ with selective repeat

6.3 Structured Sequences

- Block codes
- Convolutional codes
- Turbo codes

Channel Models

Discrete Memoryless Channel (DMC)

- The outcomes relate to the current input only and depends on the current probability for each independent transmission

$$P(Z \mid U) = \prod_{m=1}^{N} P(z_m \mid u_m) \tag{6.12}$$

Bianry Symmetric Channel

- The conditional probability for transmission (1|0) and receiving (0|1) is symmetric

$$P(0 \mid 1) = P(1 \mid 0) = p \tag{6.13}$$

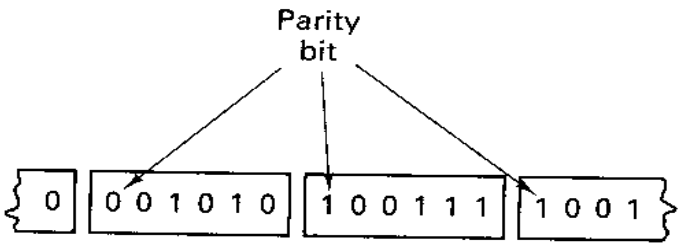
$$P(1 \mid 1) = P(0 \mid 0) = 1 - p$$

6.3.2 Code Rate and Redundancy

- In case of block codes, encoder transforms each k-bit data block into a larger block of n-bits called code bits or or channel symbol
- The $(n-k)$-bits added to each data block are are called redundant bits, parity bits or check bits
- They carry no new information
- Ratio of redundant bits to data bits: $(n-k)/k$ is called redundancy of code
- Ratio of data bits to total bits, k/n is called code rate

6.3.3 Parity-Check Codes

Single-parity-Check Code



(a)

Rectangular Code

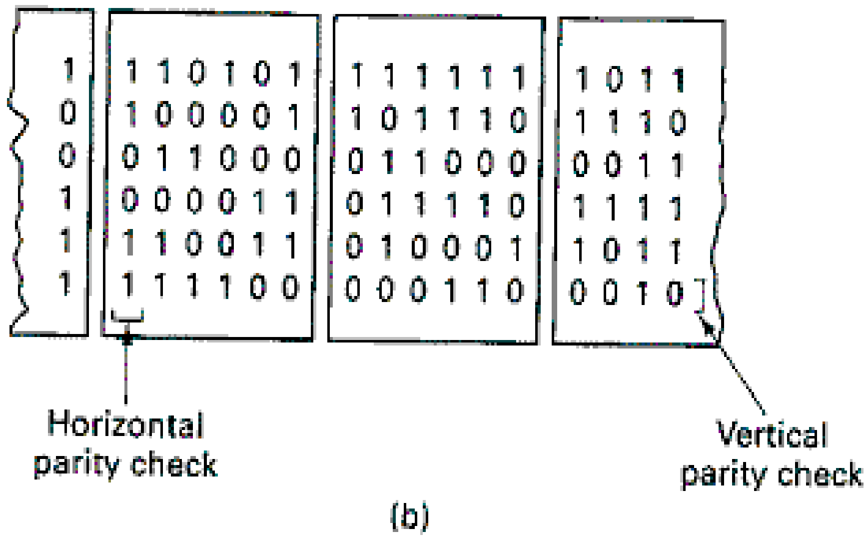


Figure 6.8: Parity checks for parallel structure

6.3.4 Why Use Error-Correction Coding

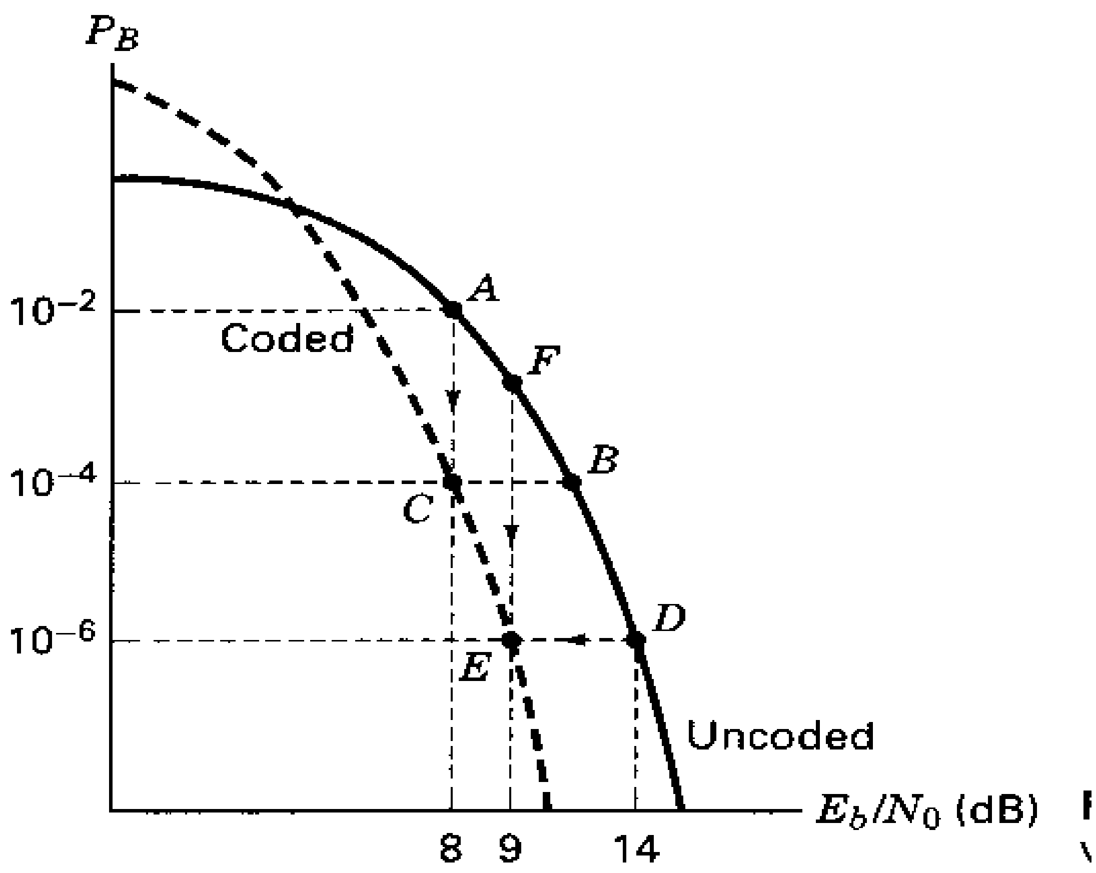


Figure 6.9:Comparison of typical coded versus uncoded error performance

- Trande-Off 1: Error Performance vs. Bandwidth
- Trande-Off 2: Power versus Bandwidth
- Coding Gain

$$(E_b/N_0)_u dB-(E_b/N_0)_c dB$$

- Trade-Off 3: Data Rate versus Bandwidth

$$(E_b/N_0)=(Pr/N_0)(1/R)$$

- Trade-Off 4: Capacity versus Bandwidth

6.4 Linear block Codes

6.4.1 Vector Spaces

- The set of all binary n-tuple is called a vector space over the binary field of 0 and 1

Addition	Multiplication
$0 \oplus 0 = 0$	$0 \cdot 0 = 0$
$0 \oplus 1 = 1$	$0 \cdot 1 = 0$
$1 \oplus 0 = 1$	$1 \cdot 0 = 0$
$1 \oplus 1 = 0$	$1 \cdot 1 = 1$

6.4.2 Vector Subspaces

- A subset S of the vector space is called a subspace if
  - The all-zeros vector is in S
  - The sum of any two vectors in S is also in S (Closure property./Linear property) e.g. { 0000 0101 1010 1111 }

6.4.3 A (6,3) Linear Block Code Examples

Message Vector	Codeword
000	000000
100	110100
010	011010
110	101110
001	101001
101	011101
011	110011
111	000111

Table 6.1: Assignment of Codewords to Messages

6.4.4 Generator Matrix

- If k is large, a lookup table implementation of the encoder becomes prohibitive
- Let the set of 2^k codewords {U} be described as:

$$\mathbf{U} = m_1\mathbf{V}_1 + m_2\mathbf{V}_2 + \ldots\ldots + m_k\mathbf{V}_k$$

- In general, generator matrix can be defined by the following k x n array:

$$G = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_k \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & & & \\ v_{k1} & v_{k2} & \cdots & v_{kn} \end{bmatrix} \tag{6.24}$$

- Generation of codeword U:

$$\mathbf{U} = \mathbf{mG} \tag{6.25}$$

Example:

- Let the generator matrix be:

$$G = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad (6.26)$$

- Generate Codeword U4 for the fourth message vector 1 1 0 in Table 6.1

U_4 =[1 1 0] V=V_1+V_2+0*V_3

= 110100+011010+000000

=101110 (Codeword for the message vector 110)

6.4.5 Systematic Linear Block codes

- A systematic (n,k) linear block code is a mapping a k-dimensional message vector to an n-dimensional code word such that part of the sequence has k message digits and remaining (n-k) are parity digits
- A systematic linear code will have a generator matrix

$$G = \left[P \;\vdots\; I_k \right] = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1,(n-k)} & 1 & 0 & \cdots & 0 \\ p_{21} & p_{22} & \cdots & p_{2,(n-k)} & 0 & 1 & \cdots & 0 \\ \vdots & & & & & & & \vdots \\ p_{k1} & p_{k2} & \cdots & p_{k,(n-k)} & 0 & 0 & \cdots & 1 \end{bmatrix} \qquad (6.27)$$

- Combining (6.26) and (6.27):

$$\mathrm{u}_1, \mathrm{u}_2, \ldots \mathrm{u}_n = [\mathrm{m}_1, \mathrm{m}_2, \ldots \mathrm{m}_k] \times \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1,(n-k)} & 1 & 0 & \cdots & 0 \\ p_{21} & p_{22} & \cdots & p_{2,(n-k)} & 0 & 1 & \cdots & 0 \\ \vdots & & & & & & & \vdots \\ p_{k1} & p_{k2} & \cdots & p_{k,(n-k)} & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Where

$u_i = m_1 p_{1i} + m_2 p_{2i} + \ldots . m_k p_{ki}$ for $i=1,\ldots(n-k)$

$= m_{i-n+k}$ for $i=(n-k+1)\ldots . n$

- The systematic code vector can be expressed as:

$$U = \underbrace{p_1, p_2, \ldots, p_{n-k}}_{parity\ \ bits}, \underbrace{m_1, m_2, \ldots, m_k}_{message\ \ bits} \qquad (6.28)$$

$$\begin{aligned}
p_1 &= m_1 p_{11} + m_2 p_{21} + \cdots + m_k p_{k1} \\
p_2 &= m_1 p_{12} + m_2 p_{22} + \cdots + m_k p_{k2} \\
p_{n-k} &= m_1 p_{1,(n-k)} + m_2 p_{2,(n-k)} + \cdots + m_k p_{k,(n-k)}
\end{aligned} \qquad (6.29)$$

Example

- For (6,3) code example in sec.6.4.3, the codewords can be described as:

$$U = [m_1, m_2, m_3] \left[\begin{array}{ccc} \underbrace{\begin{matrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{matrix}}_{P} & \vdots & \underbrace{\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}}_{I_3} \end{array} \right] \tag{6.30}$$

$$= \underbrace{m_1 + m_3}_{u_1}, \underbrace{m_1 + m_2}_{u_2}, \underbrace{m_2 + m_3}_{u_3}, \underbrace{m_1}_{u_4}, \underbrace{m_2}_{u_5}, \underbrace{m_3}_{u_6} \tag{6.31}$$

6.4.6 Parity-Check Matrix

- Let H denote the parity check matrix, that will enable us to decode the received vectors
- For each (k x n) generator matrix G, there exists an (n-k)x n matrix H, such that rows of G are orthogonal to the rows of H: $GH^T=0$
- Fulfilling the orthogonality requirements:

$$H = \left[I_{n-k} \vdots P^T \right] \tag{6.32}$$

And

$$H^T = \begin{bmatrix} I_{n-k} \\ \hline P \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & & \\ 0 & 0 & \cdots & 1 \\ p_{11} & p_{12} & \cdots & p_{1,(n-k)} \\ p_{21} & p_{22} & \cdots & p_{2,(n-k)} \\ \vdots & & & \\ p_{k1} & p_{k2} & \cdots & p_{k,(n-k)} \end{bmatrix} \tag{6.33}$$

- It is easy to verify from here:

$$UH^T = p_1 + p_2 + \ldots P_{n-k} = 0$$

Where U is a code word generated by matrix G iff $UH^T = 0$

6.4.7 Syndrome Testing

- Let r be received vector where U vector was transmitted :

$$\mathbf{r = U + e} \qquad (6.34)$$

- The syndrome of r is defined as:

$$\mathbf{S = rH^T} \qquad (6.35)$$

- Combining (6.34) and (6.35)

$$\mathbf{S = (U + e)H^T = U\,H^T + e\,H^T} \qquad (6.36)$$

$$\mathbf{S = e\,H^T} \qquad (6.37)$$

- Requirements of the parity-check matrix
  - No column of H can be all zeros, or else an error in the corresponding codeword position would not affect the syndrome and would be undetectable
  - All columns of H must be unique. If two columns of H were identical, errors in these two corresponding codeword positions would be indistinguishable

Example

- Codeword U = 1 0 1 1 1 0 , and r = 0 0 1 1 1 0 Find S=rH^T

$$S = rH^T$$

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

$$= \begin{bmatrix} 1, & 1+1, & 1+1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$$S = eH^T$$

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

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

6.4.8 Error Correction

- Arranging 2n n-tuples; representing possible received vectors, in an array is called standard array. Standard array for (n,k) code is:

$$
\begin{array}{cccccc}
U_1 & U_2 & \cdots & U_i & \cdots & U_{2^k} \\
e_2 & U_2 + e_2 & \cdots & U_i + e_2 & \cdots & U_{2^k} + e_2 \\
e_3 & U_2 + e_3 & \cdots & U_i + e_3 & \cdots & U_{2^k} + e_3 \\
\vdots & \vdots & & \vdots & & \\
e_j & U_2 + e_j & \cdots & U_i + e_j & \cdots & U_{2^k} + e_j \\
\vdots & \vdots & & \vdots & & \\
e_{2^{n-k}} & U_2 + e_{2^{n-k}} & \cdots & U_i + e_{2^{n-k}} & \cdots & U_{2^k} + e_{2^{n-k}}
\end{array}
\tag{6.38}
$$

- Each row, called a coset consists of an error pattern in the first column called coset leader
- If error pattern is not a coset leader, erroneous decoding will result

The syndrome of a Coset

- Coset is a short for “a set of numbers having a common feature”
- If e_j is the coset leader then U_i+e_j is an n-tuple in this coset.
- Syndrome of this n-tuple is:

$$\mathbf{S = (U_i + e_j)\, H^T = e_j\, H^T}$$

- The syndrome must be unique to estimate the error pattern

Error Correction Decoding

- Calculate the syndrome of R using $S=rH^T$
- Locate the coset leader (error pattern) e_j, whose syndrome equals $\mathbf{rH}^T$
- This error pattern is assumed to be the corruption caused by the channel
- The corrected received vector, or code word, is identified as $U=r+e_j$. We retrieve the valid codeword by subtracting(adding) the identified error

Locating the Error Pattern

- Example of a standard array for a (6,3) code is shown:

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

- Computing $e_j H^T$ for each coset leader

$$S = e_j \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

- The results are:

Error Pattern	Syndrome
000000	000
000001	101
000010	011
000100	110
001000	001
010000	010
100000	100
010001	111

Table 6.2: Syndrome lookup Table

Error Correction Example

- Error pattern is an estimate of error, the decoder addes the estimated error to received signal to obtain an estimate of transmitted code word as:

$$\hat{U} = r + \hat{e} = (U + e) + \hat{e} = U + (e + \hat{e}) \qquad (6.40)$$

- Example: let U=101110, and r=001110, then show how the decoder can correct the error using look-up table 6.2

$$S = [0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0]H^T = [1 \quad 0 \quad 0]$$

estimated error :

$$\hat{e} = 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0$$

The corrected vector is estimated by :

$$\begin{aligned} \hat{U} &= r + \hat{e} \\ &= 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 + 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \\ &= 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \end{aligned}$$

6.5 Error Detection and Correcting Capability

6.5.1 Weigh and Distance of Binary Vector

- Hamming distance between two codewords is the number of elements in which they differ
- Hamming weight is the number of nonzero elements

Example:

$$U = 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1$$

$$V = 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0$$

$$w(U) = 5$$

$$d(U,V) = w(U+V) = 6$$

Definitions:

Minimum Distance of a Linear Code: The minimum distance among all the distances between each pair of codes in the code set

- The error-correcting capability t of a code: the maximum number of guaranteed correctable errors per codeword

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$$

- Error detecting capability defined in terms of d_{min}

$$e = d_{\min} - 1$$

7.1 CONVOLUTIONAL ENCODING

- A convolutional code is described by three integers, n, k, and K where the ratio k/n is called the rate of the code

- The integer K is constraint length; it represents number of k-tuple stages in the encoding shift register.

- Encoder has memory—the n-tuple emitted by the convolutional encoding procedure is not only a function of an input k-tuple, but is also a function of the previous K-1 input k-tuples

Block Diagram of a Typical Communication Link

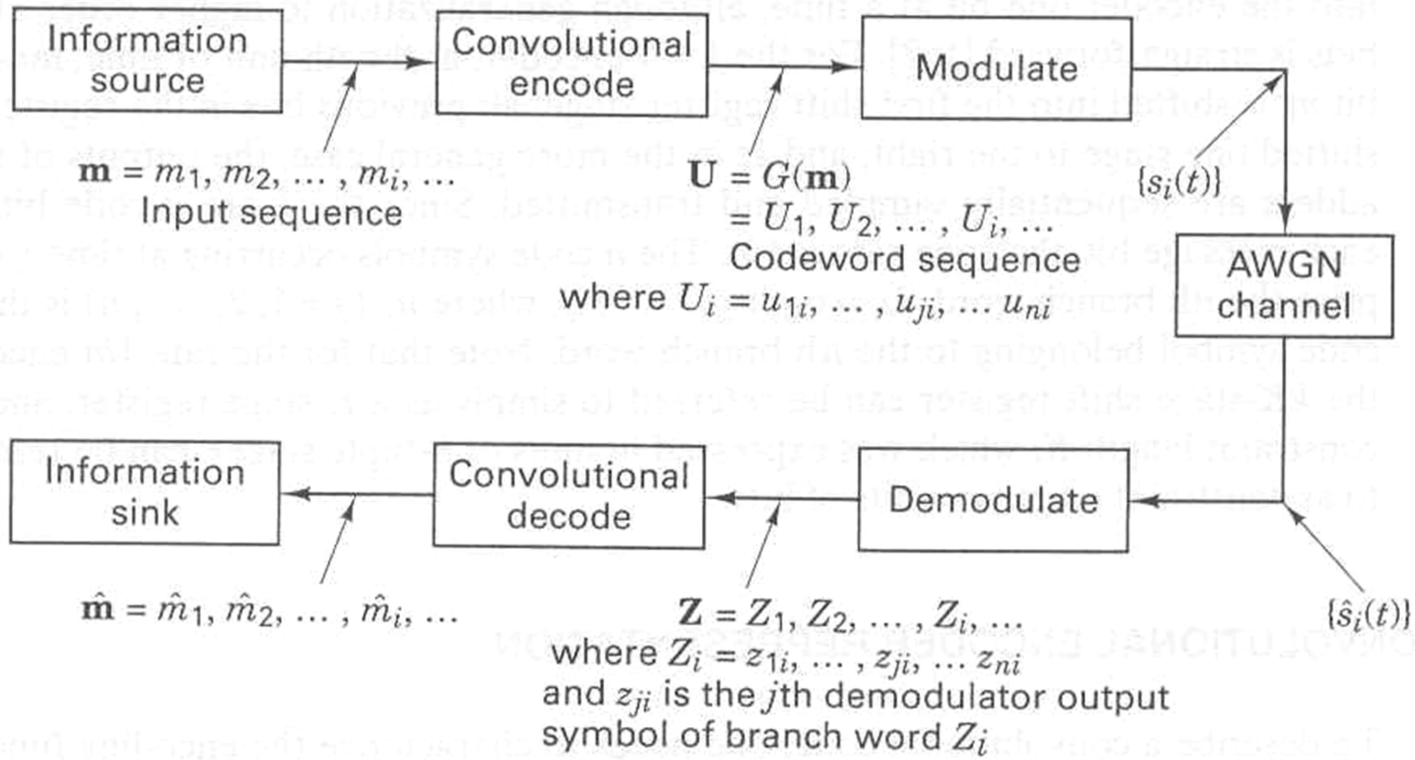


Figure 7.1: Encode/decode and modulate/demodulate portions of a communication link

7.2.1 Connection Representation

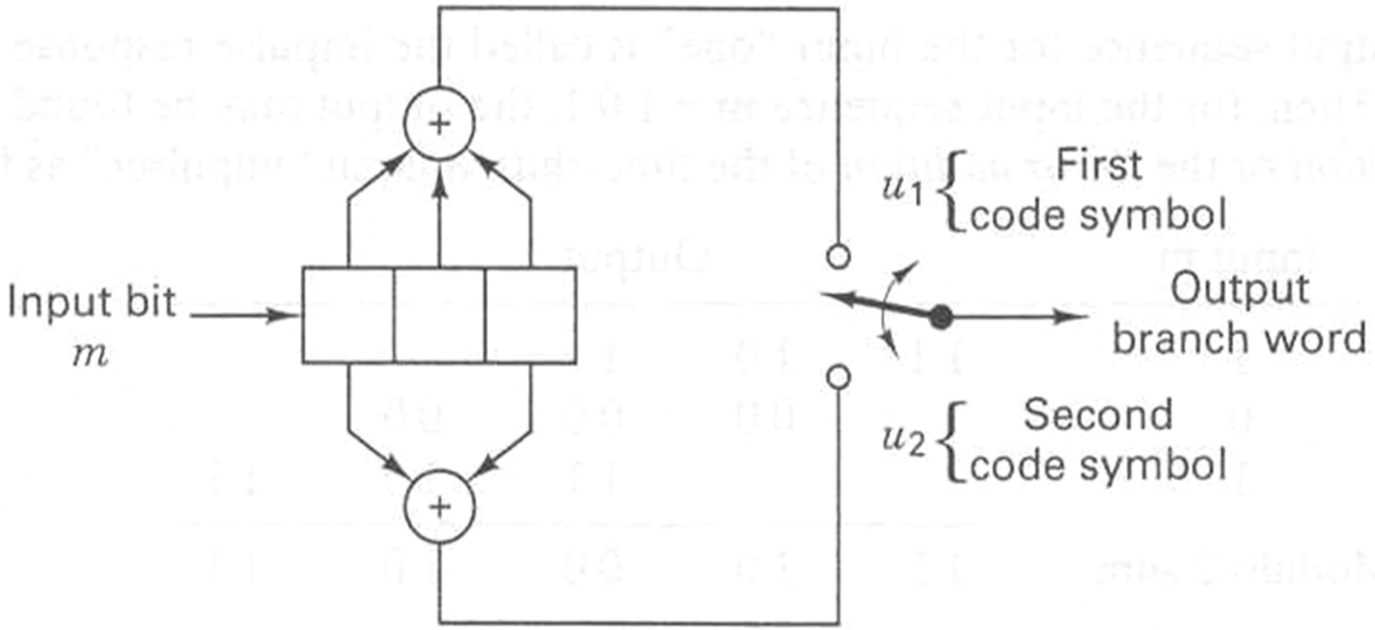


Figure 7.3: Convolutional Encoder (rate ½, k=3)

7.2.1.1 Impulse Response of the Encoder

- One approach for the encoder could be by using impulse response, response of the encoder to a single “one” bit that moves through it
- Consider figure 7.3:
- Input sequence : 1 0 0
- Output sequence: 11 10 11
- Output maybe found by the superposition or linear addition of time-shifted input impulses:

Register Contents	Branch Word U1	U2
100	1	1
010	1	0
001	1	1

Input m	Output				
1	11	10	11		
0		00	00	00	
0			00	00	00
Modulo-2 sum	11	10	11	00	00

- We can write the connection vector g1 for the upper connections and g2 for the lower connections as:

$$g_1 = 111 \quad g_2 = 101$$

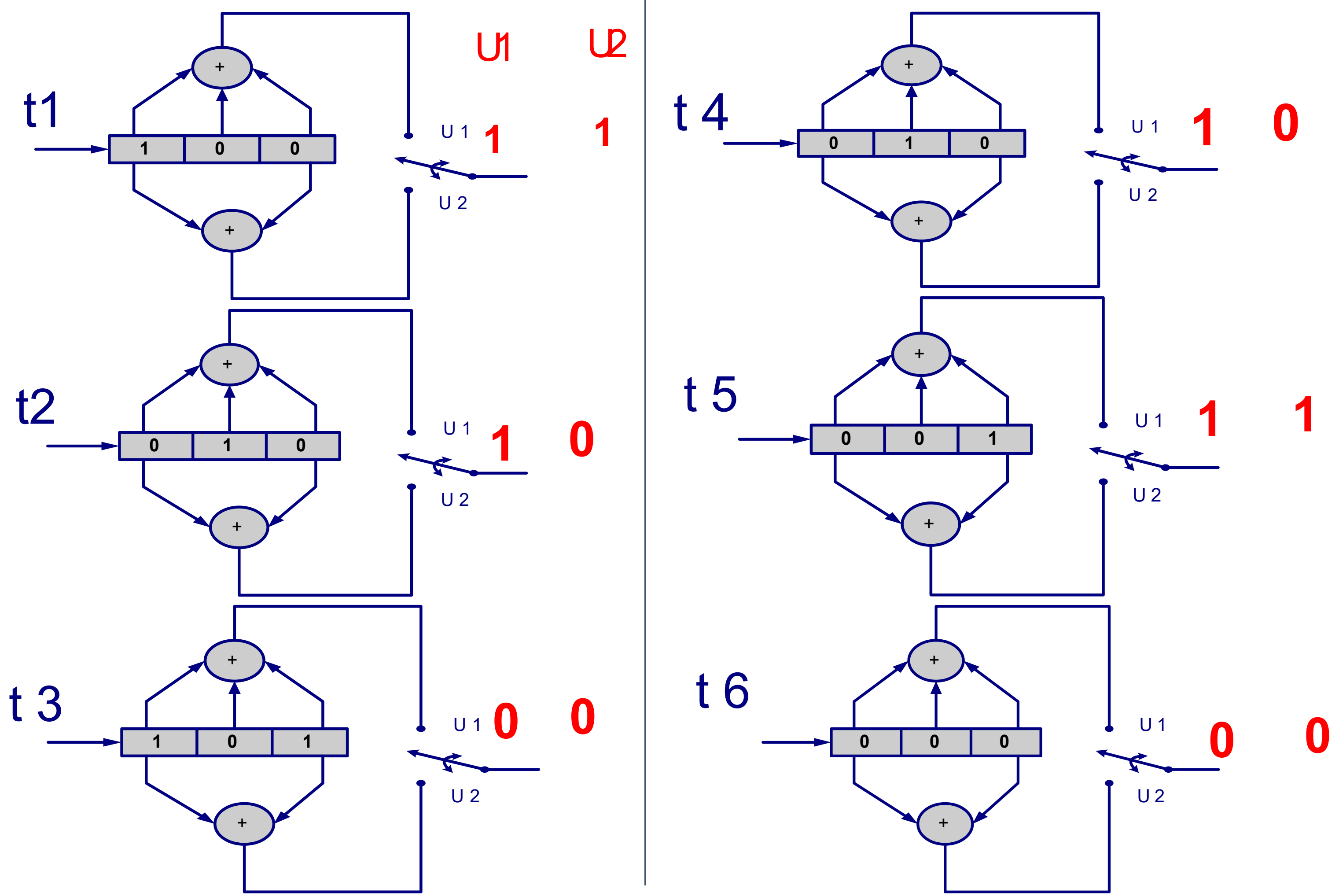
- Let m=101 be convolutionally encoded with the encoder as:

$$m=101 \longrightarrow \boxed{\text{Encoder}} \longrightarrow U$$

- Convolutionally encoding a message sequence with rate ½, k=3 encoder is illustrated:

Input m	Output				
1	11	10	11		
0		00	00	00	
1			11	10	11
Modulo-2 sum	11	10	00	10	11

Time Encoder Output | Time Encoder Output



Output Sequence: 11 10 00 10 11 00

7.2.1.2 Polynomial Representation

- Convolutional encoder maybe represented with a set of n generator polynomials, one for each modulo-2 adders
- Continuing with the same example, we can write the generator polynomial for upper connections $g_1(X)$ and $g_2(X)$ for lower connections:

$$g_1(X) = 1 + X + X^2$$

$$g_2(X) = 1 + X^2$$

- U(X) is the output sequence

$$U(X) = m(X)g_1(X) \quad \text{interlaced} \quad \text{with} \quad m(X)g_2(X)$$

- Where m= 101,encoder can be found as:

$$m(X)g_1(X) = (1+X^2)(1+X+X^2) = 1+X+X^3+X^4$$

$$m(x)g_2(X) = (1+X^2)(1+X^2) = 1+X^4$$

$$m(X)g_1(X) = 1+ \quad X + 0X^2 + \quad X^3 + X^4$$

$$m(X)g_2(X) = 1+0X+0X^2+0X^3+X^4$$

$$U(X) = (1,1) + (1,0)X + (0,0)X^2 + (1,0)X^3 + (1,1)X^4$$

$$U = 1 \;\; 1 \qquad 1 \;\; 0 \qquad 0 \;\; 0 \qquad 1 \;\; 0 \qquad 1 \;\; 1$$

Example 7.1 Convolutional Encoding

- For the encoder shown in figure 7.3, show state changes and resulting codeword sequence U for m = 1 1 0 1 1 followed by 2 zeros to flush the registers.
- Assume initial contents of the register are all zeros

Input bit m_i	Register Contents	State at time t_i	State at t_i+1	Branch Word at t_i U1 U2
----	000	00	00	----
1	100	00	10	1 1
1	110	10	11	0 1
0	011	11	01	0 1
1	101	01	10	0 0
1	110	10	11	0 1
0	011	11	01	0 1
0	0 0 1	01	00	1 1

State t_i+1 State t_i

Output Sequence: 11 01 01 00 01 01 11

7.2.2 State Representation and State Diagram

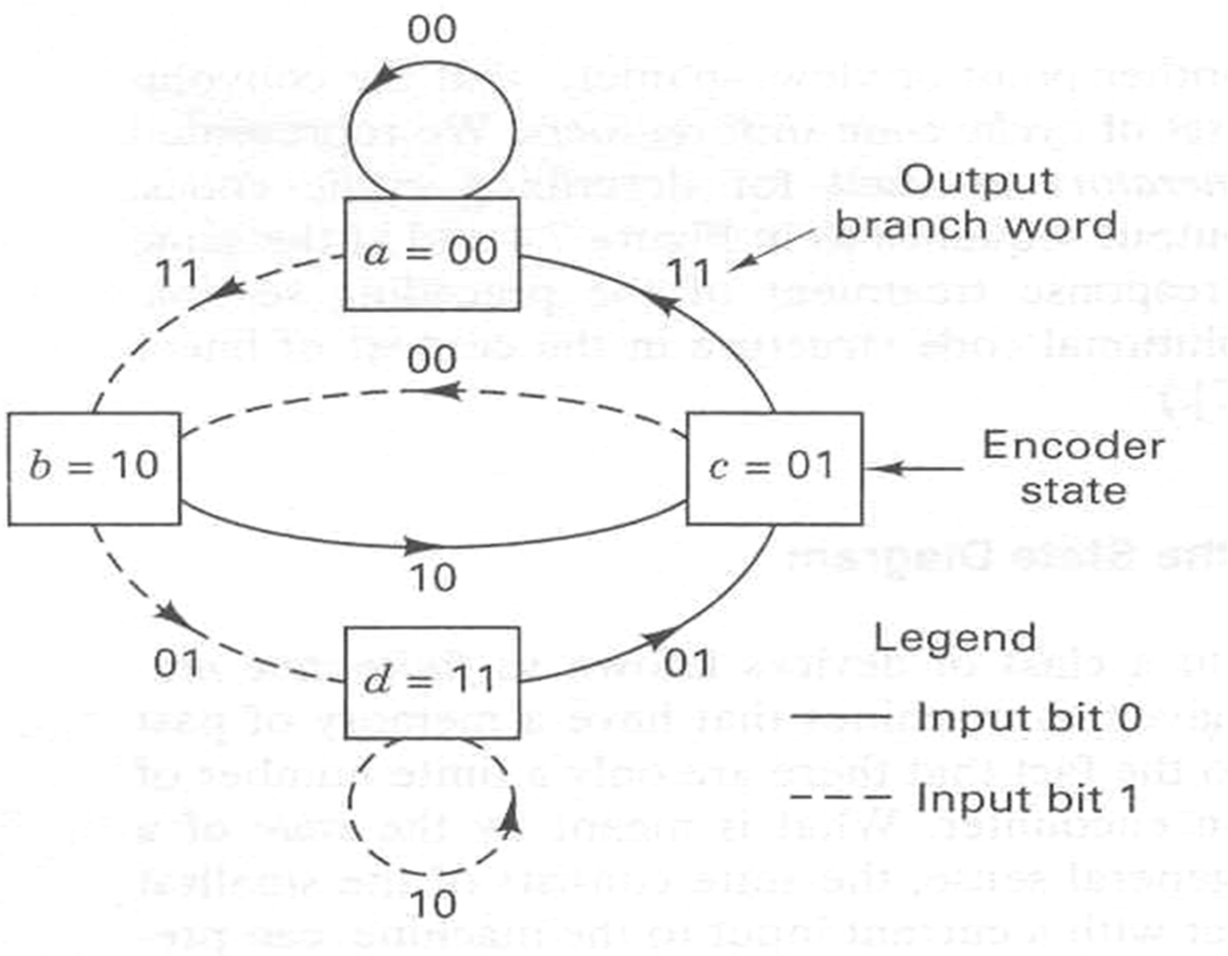


Figure 7.5: Encoder state (the contents of the K-1 leftmost registers) diagram (rate ½ ,k=3)

7.2.3 The Tree Diagram

- Tree diagram adds the dimension of time to the state diagram

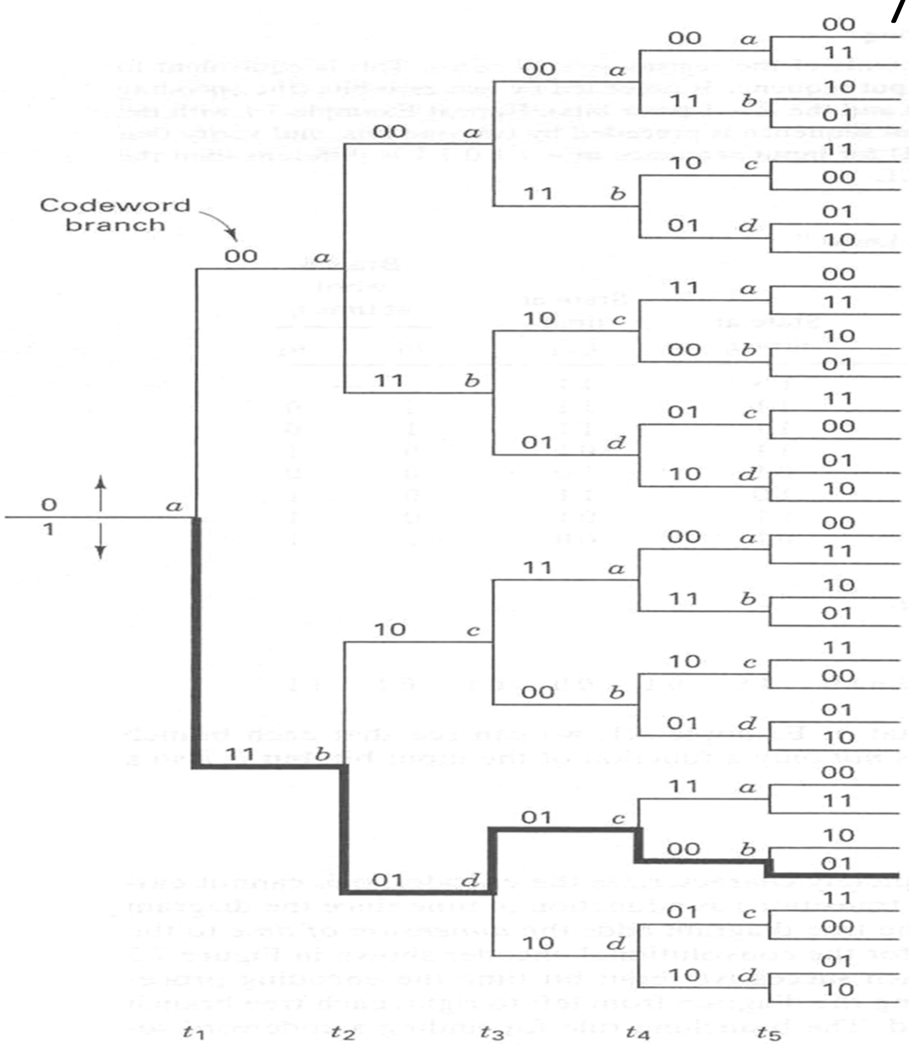
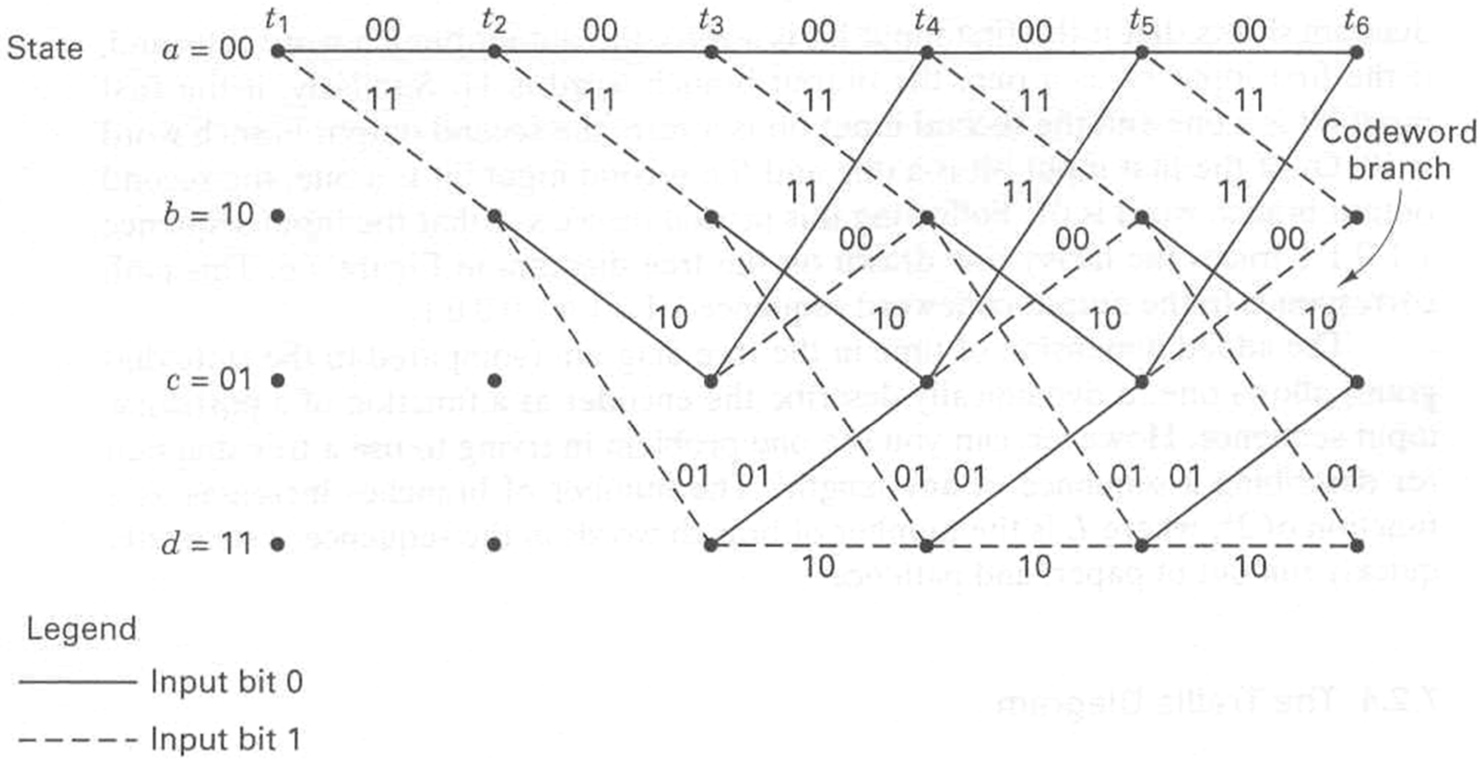


Figure 7.6: Tree representation of encoder (rate ½, k=3)

7.2.4 The Trellis Diagram



■The trellis diagram, by exploiting the repetitive structure, provides a more manageable encoder description

Figure 7.7: Encoder trellis diagram (rate ½, K=3)

7.3.3 The Viterbi convolutional decoding Algorithm

Input data sequence	**m:**	1	1	0	1	1	...
Transmitted codeword	**U:**	11	01	01	00	01	...
Received sequence	**Z:**	11	01	01	10	01	...

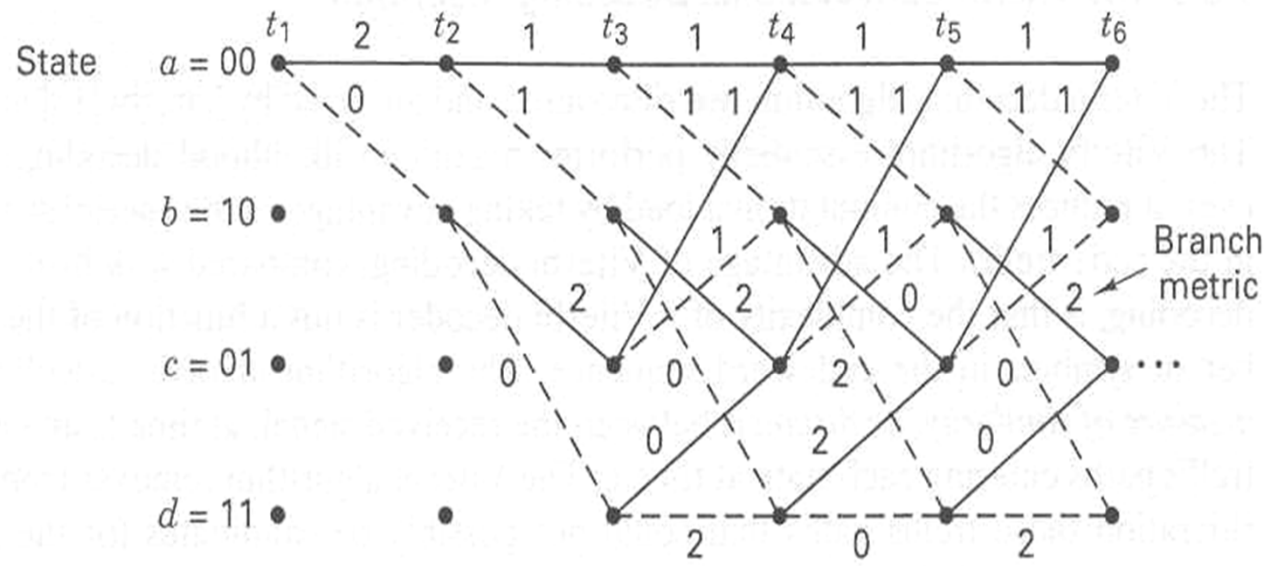


Figure 7.10: Decoder trellis diagram (rate ½ , K=3)

Steps in Decoding

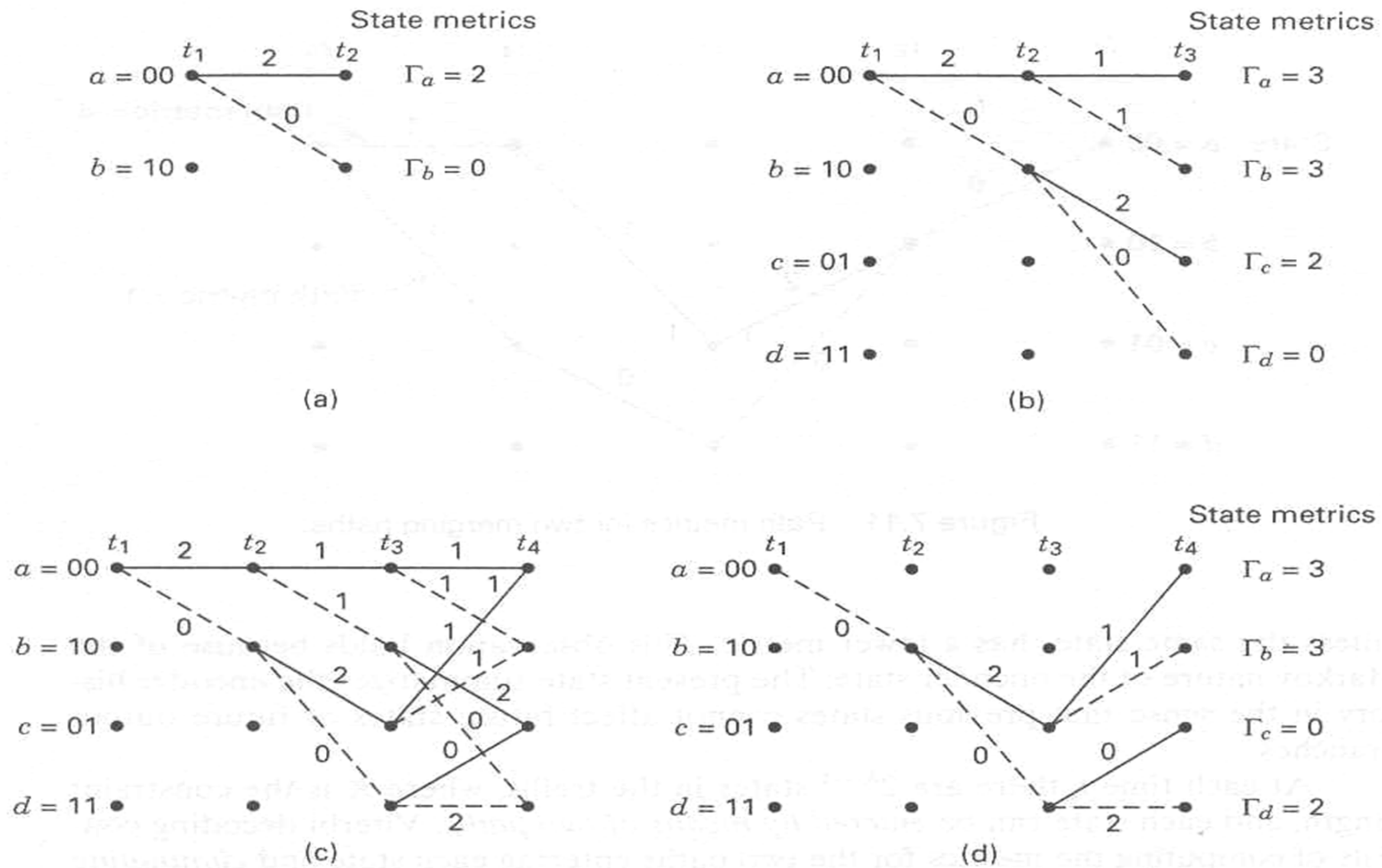


Figure 7.12: selection of survivors (a) survivors at t2 (b) survivors at t3 (c) metric comparison at t4 (d) survivors at t4

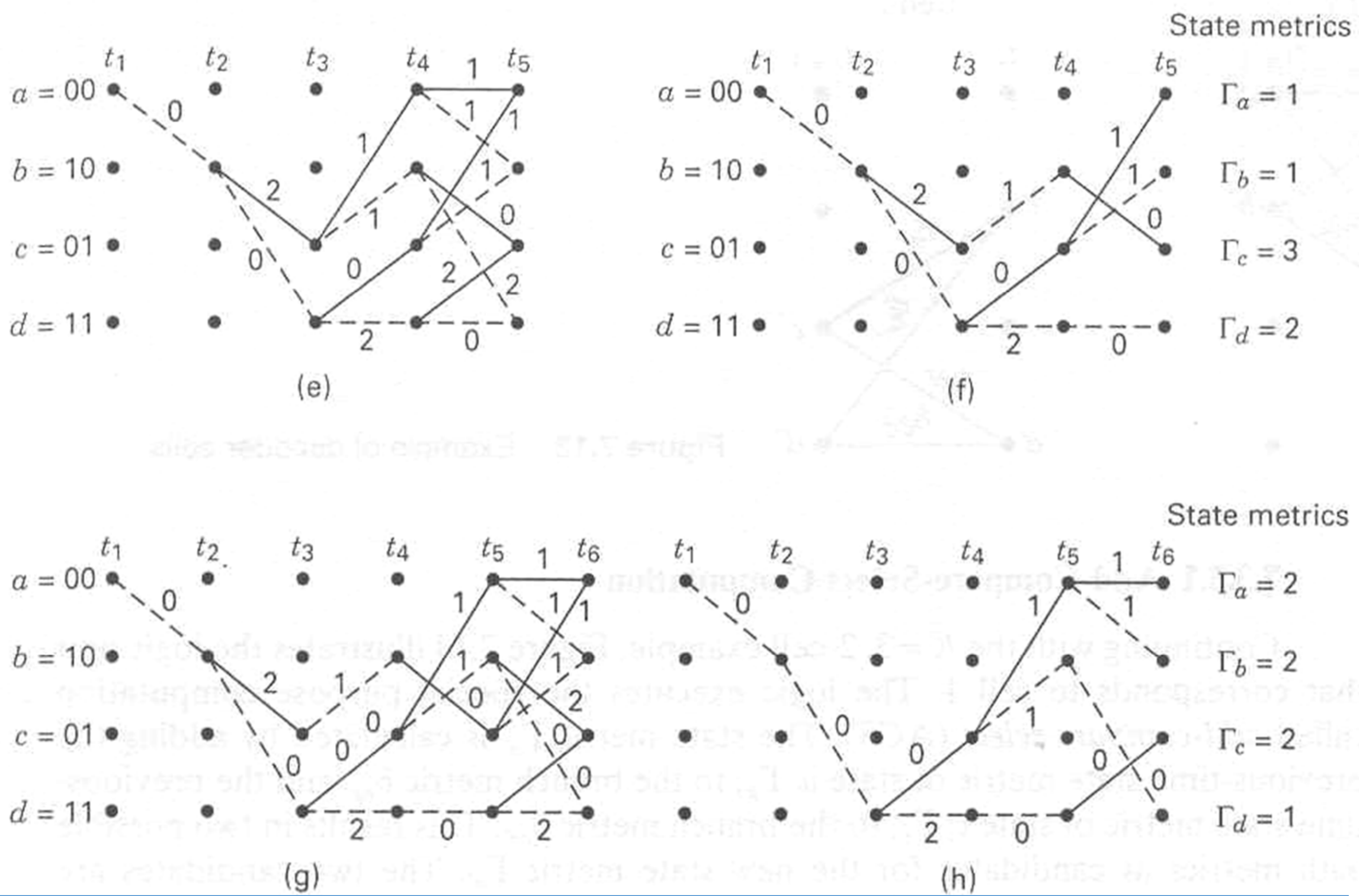


Figure 7.12: selection of survivors (e) metric comparison at t5 (f) survivors at t5 (g) metric comparison at t6 (h) survivors at t6

6. Best Known Convolution Codes

Rate	Constraint Length	Free Distance	Code Vector
$\frac{1}{2}$	3	5	111 101
$\frac{1}{2}$	4	6	1111 1011
$\frac{1}{2}$	5	7	10111 11001
$\frac{1}{2}$	6	8	101111 110101
$\frac{1}{2}$	7	10	1001111 1101101
$\frac{1}{2}$	8	10	10011111 11100101
$\frac{1}{2}$	9	12	110101111 100011101
$\frac{1}{3}$	3	8	111 111 101
$\frac{1}{3}$	4	10	1111 1011 1101
$\frac{1}{3}$	5	12	11111 11011 10101
$\frac{1}{3}$	6	13	10111 110101 111001
$\frac{1}{3}$	7	15	1001111 1010111 1101101
$\frac{1}{3}$	8	16	11101111 10011011 10101001

Table 7.4: Optimum Short Constraint Length Convolutional Codes