2 CODIFICAÇÃO DE CANAL: CÓDIGOS DE BLOCO LINEARES

Um canal de comunicação pode apresentar uma série de imperfeições que dificultam a correta interpretação e perfeita reprodução dos sinais transmitidos. Tais imperfeições se apresentam como ruídos, distorções, interferências, desvanecimentos, etc., e, como consequência, a função do receptor pode ser resumida como sendo habilidade de apresentar em sua saída a *melhor estimativa* da informação ou mensagem que foi transmitida [1].

Em sistemas de comunicações digitais, o parâmetro de desempenho que permite quantificar o que pode ser a *melhor estimativa* é a probabilidade de erro de bit, P_b, cujo valor, para ser considerado satisfatório, depende fundamentalmente do tipo de informação que está sendo transmitida. A questão fundamental apresentada neste capítulo está diretamente relacionada com o controle da probabilidade de erro de bit, abordada a partir da próxima seção.

2.1. INTRODUÇÃO À CODIFICAÇÃO DE CANAL

A codificação de canal é um processo em que redundâncias são introduzidas antes da transmissão, com o objetivo de permitir que, no receptor, a semelhança entre o sinal que foi transmitido e o sinal que foi reproduzido seja a máxima possível. Ou, por outro lado, é um processo que permite a redução da P_b a valores tão baixos quanto possíveis. A partir desse ponto é inevitável a imposição de uma questão: Em termos objetivos, o que se pode esperar obter como *máxima semelhança* ou P_b *tão baixa quanto possível* com a codificação de canal? Essa questão foi parcialmente respondida no capítulo anterior através do Teorema da Codificação de Canal, cuja consequência é reproduzida a seguir por conveniência [1][2][3].

Desde que a taxa de transmissão seja menor do que a capacidade do canal, então existe um esquema de codificação capaz de permitir a obtenção de taxas de erro de bit arbitrariamente baixas.

O Teorema da Codificação de Canal, no entanto, é insatisfatório sob o ponto de vista prático porque não conduz a uma indicação do esquema de codificação capaz de produzir um resultado esperado, nem tampouco, o seu grau de complexidade ou da dificuldade de encontrá-lo. Uma discussão mais detalhada sobre esse assunto pode ser encontrada em diversas publicações sobre Teoria da Informação.

De fato, a busca de um esquema de codificação, que em geral é um processo heurístico, nem sempre tem como principal meta alcançar desempenhos próximos dos apresentados pelos limites fundamentais da Teoria da Informação. Em sistemas reais, a busca de um esquema de codificação pode estar associada a aspectos práticos como velocidade de processamento, complexidade de implementação, etc.

Independentemente de qual seja a abordagem utilizada na busca de um esquema de codificação, é necessário um bom conhecimento dos fundamentos associados às técnicas de codificação para controle de erro.

O objetivo deste capítulo é apresentar os fundamentos dos Códigos de Bloco Lineares e seus desempenhos. Eles são apresentados em cinco seções:

- Definições Fundamentais
- Códigos de Bloco Lineares
- Desempenho dos Códigos de Bloco Lineares
- Códigos Cíclicos
- Características dos Códigos de Bloco Bem Conhecidos

2.2. CÓDIGOS DE BLOCO: DEFINIÇÕES INICIAIS

Antes da apresentação dos Códigos de Blocos Lineares, algumas definições iniciais são oportunas. Códigos de blocos se caracterizam pelo fato do processo de codificação ser feito sobre blocos de bits ou bloco de símbolos. Isso quer dizer que um feixe de bits ou símbolos é segmentado em blocos de k bits ou símbolos, a partir dos quais são geradas palavras códigos com n bits ou símbolos. Assim, a notação que caracteriza um código de bloco é (n, k). Por conveniência, a partir deste ponto a notação (n, k) estará associada à quantidade de bits. Quando a notação (n, k) for usada para representar símbolos, isso será definido explicitamente.

Se k bits estão contidos em um bloco de n bits, então a quantidade de bits de redundância introduzidos no processo de codificação é $(n - k)$.

2.2.1. TAXA DE CODIFICAÇÃO

A taxa de codificação de um código de bloco é definida como sendo a relação entre o número de bits de informação e o número de bits da palavra código. Ou seja,

$$R_c = \frac{k}{n}. \tag{2.1}$$

A taxa de codificação é uma indicação relativa de quantos bits de informação são transmitidos por palavra código. Uma vez que $0 < k \leq n$, então $0 < R_c \leq 1$. Entretanto, para que um código produza algum benefício, é necessário que $k < n$, ou $(n - k) > 0$. Consequentemente, $0 < R_c < 1$.

Nota-se que, se nenhum artifício for usado para compensar o acréscimo de bits devido à introdução da redundância, então, para a manutenção da taxa de transmissão dos bits de informação é necessário aumentar a taxa de transmissão total, resultando em um acréscimo ou expansão da largura de faixa. Essa expansão da largura de faixa é de exatamente a $1/R_c$. Ou seja, quanto maior for o número de bits de redundância introduzidos, maior será a expansão da largura de faixa.

2.2.2. GANHO DE CODIFICAÇÃO

O benefício obtido com o processo de codificação pode ser quantificado por meio do *ganho de codificação*. O ganho de codificação é definido como sendo a relação entre E_b/N_0 do sinal não codificado, pelo E_b/N_0 do sinal codificado, para uma dada taxa de erro, i.e. o ganho de codificação é

tipicamente uma função de P_b. A expressão do ganho de codificação, em dB, é apresentada a seguir [3][4][6].

$$G = 10\log\left(\frac{E_b}{N_0}\right)_{nc} - 10\log\left(\frac{E_b}{N_0}\right)_c \quad \text{(dB)} \tag{2.2}$$

Onde, $(E_b/N_0)_{nc}$ é a relação entre a energia de bit e a densidade espectral de ruído sem codificação e $(E_b/N_0)_c$ é a relação entre a energia de bit e a densidade espectral de ruído com codificação.

As curvas características de P_b em função de E_b/N_0, para um esquema de transmissão codificado e não codificado, são apresentado na Figura 2.1. Um cuidado deve ser tomado para a correta determinação de P_b em função de E_b/N_0 com a codificação: o valor de E_b refere-se à energia por bit de informação, ou seja, admitindo-se que a energia total gasta para a transmissão seja a mesma para os dois casos, então a energia por bit de informação com a codificação é menor do que a energia de bit sem a codificação, devido à inserção dos bits de redundância. Assim sendo, a relação entre $(E_b/N_0)_c$ e $(E_b/N_0)_{nc}$ fica afetada pela taxa de codificação na forma

$$\left(\frac{E_b}{N_0}\right)_c = R\left(\frac{E_b}{N_0}\right)_{nc} \tag{2.3}$$

Os valores de taxa de erro e de E_b/N_0 apresentados na Figura 2.1 referem-se a um esquema hipotético e tem por objetivo permitir generalizar conclusões apresentadas a seguir.

1) Para baixos valores de E_b/N_0 a codificação não apresenta nenhum benefício, ou seja, o ganho de codificação pode ser nulo, para o valor de E_b/N_0 determinado pelo cruzamento das curvas, ou negativo para valores menores.
2) Para diferentes valores de P_b obtém-se diferentes ganhos de codificação. Por exemplo, para $P_b = 10^{-4}$ o ganho de codificação é igual o a 1,4 dB, enquanto para $P_b = 10^{-6}$ o ganho sobe para 2 dB. Isso demonstra a dependência do ganho de codificação com P_b.
3) A codificação permite obter redução de P_b com a mesma energia (E_b/N_0 constante) em relação ao sinal sem codificação. Por exemplo, para $E_b/N_0 = 7$ dB, a P_b cai de 10^{-3} para um pouco mais que 10^{-5}.

Considerando-se a expansão de largura de faixa provocada pela codificação, pode-se concluir ainda:

4) A diminuição de P_b e/ou E_b/N_0 decorrente da codificação produz uma expansão na largura de faixa.

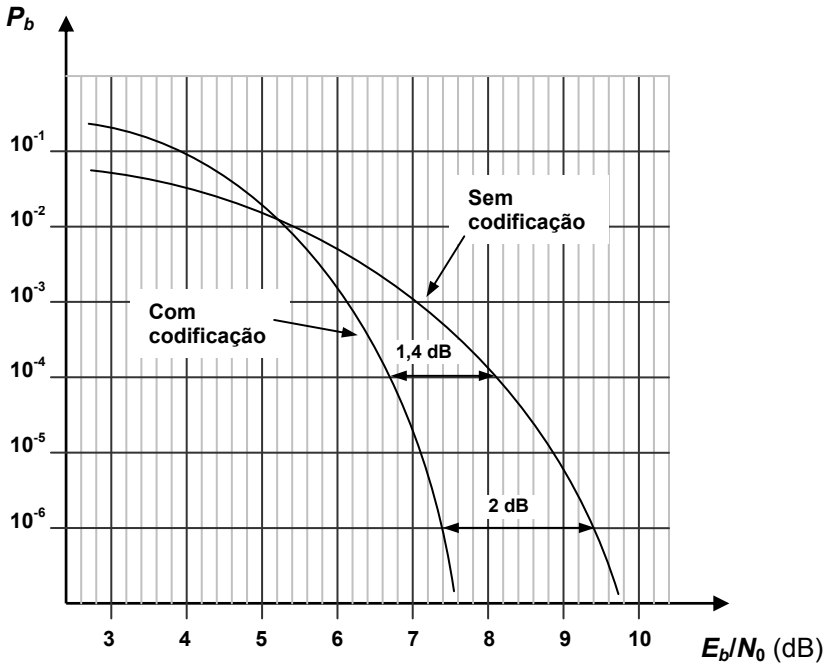

Figura 2.1 - Curvas típicas de probabilidade de erro versus E_b/N_0 para um sinal com codificação e sem codificação.

2.2.3. VETOR CÓDIGO, VETOR ERRO, VETOR RECEBIDO E VETOR DECODIFICADO

Conforme definido previamente, o processo de codificação por bloco consiste em transformar um segmento da mensagem, **m**, com *k* bits em uma palavra código ou vetor código, **c**, com *n* bits. O vetor código é transmitido e pode sofrer alterações devido às degradações impostas pelo meio de transmissão. As alterações sofridas pelo vetor códigos podem ser representadas por meio de um vetor erro, **e**. Um vetor código, **c**, somado com um vetor erro, **e**, resulta em um vetor recebido, **r**. Ou seja,

$$\mathbf{r} = \mathbf{c} \oplus \mathbf{e} \,. \tag{2.4}$$

O vetor recebido é entregue ao decodificador cuja finalidade é transformar o vetor recebido no vetor decodificado, que consiste na melhor estimativa do vetor código transmitido. A partir do vetor código estimado, **c**', a melhor estimativa da mensagem, **m**', é reproduzida na saída do decodificador. Essa cadeia de transformações é ilustrada no diagrama em blocos apresentado na Figura 2.2 e pelo Exemplo 2.1.

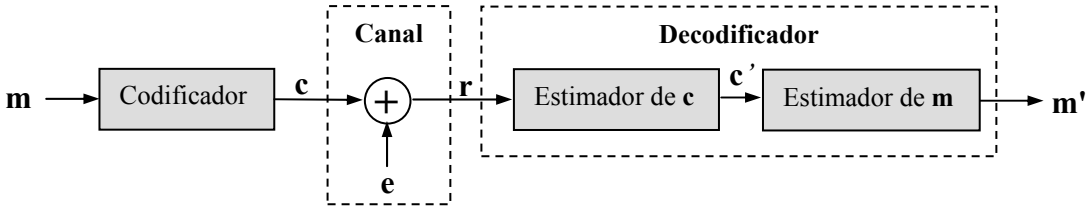

Figura 2.2. Diagrama em blocos do processo de codificação e decodificação.

EXEMPLO 2.1

Seja um código de repetição (5, 1) aplicado ao vetor mensagem **m** = 1. Admitindo que o canal introduza um vetor erro, **e** = 01010, ao vetor código, pede-se determinar todas as transformações vetoriais desde a codificação da mensagem até a obtenção da mensagem estimada na saída do decodificador.

Solução:

Evidentemente para o código de repetição (5, 1) só existem duas palavras códigos possíveis: 00000 e 11111. A palavra código correspondente à mensagem **m** = 1 é

$$\mathbf{c} = 11111$$

O vetor recebido é

$$\mathbf{r} = \mathbf{c} \oplus \mathbf{e} = 11111 \oplus 01010$$

$$\mathbf{r} = 10101.$$

Como um código de repetição pode ser decodificado por lógica majoritária, então a melhor estimativa para o vetor código a partir do vetor recebido é

$$\mathbf{c'} = 11111,$$

que resulta na mensagem estimada

$$\mathbf{m'} = 1.$$

* * *

2.2.4. PESO DE HAMMING E DISTÂNCIA DE HAMMING

O peso de Hamming de um vetor **v,** cuja notação é $w(\mathbf{v})$, é definido como o sendo o número de elementos não zero em **v**. Para um vetor binário, o peso de Hamming é igual ao número de dígitos "1" contidos em **v** [1][2][3][4][5][6].

EXEMPLO 2.2

Determinar o peso de Hamming do vetor **v** = 10110.

Solução:

$$w(\mathbf{v}) = 3.$$

* * *

A distância de Hamming entre dois vetores códigos **v** e **x**, cuja notação é $d(\mathbf{v}, \mathbf{x})$, é definida como sendo o número de posições em que os dígitos dos dois vetores que são diferentes entre si. Para o caso binário, a distância de Hamming pode ser determinada facilmente através da propriedade de adição módulo-2, pois ela é igual ao número de dígitos "1" contidos no vetor resultante da operação $\mathbf{v} \oplus \mathbf{x}$. Ou seja,

$$d(\mathbf{v}, \mathbf{x}) = w(\mathbf{v} \oplus \mathbf{x}) . \tag{2.5}$$

EXEMPLO 2.3

Determinar a distância de Hamming entre o vetor **v** = 10110 e **x** = 10101.

Solução:

$$d(\mathbf{v}, \mathbf{x}) = w(\mathbf{v} \oplus \mathbf{x}) = w(10110 \oplus 10101) = w(00011)$$

$$d(\mathbf{v}, \mathbf{x}) = 2$$

* * *

2.2.5. ESPAÇO VETORIAL E SUBESPAÇO VETORIAL

Considere um conjunto V, constituídos por K vetores $\mathbf{v_0}$, $\mathbf{v_1}$, $\mathbf{v_2}$, ... , $\mathbf{v_{K-1}}$, formado por n elementos de {0, 1}. Admita que sobre este conjunto sejam definidas duas operações, cujas regras são apresentadas na Tabela 2.1. A adição, representada por $\oplus$, definida entre os elementos de V e a multiplicação, representada por $\bullet$, entre um elemento de {0, 1} e qualquer vetor de V [1][2][3][4][5][6].

Tabela 2.1 - Operações algébricas no campo binário.

Adição	Multiplicação
$0 \oplus 0 = 0$	$0 \cdot 0 = 0$
$0 \oplus 1 = 1$	$0 \cdot 1 = 0$
$1 \oplus 0 = 1$	$1 \cdot 0 = 0$
$1 \oplus 1 = 0$	$1 \cdot 1 = 1$

O conjunto V é definido como um *espaço vetorial* sobre {0, 1} se as seguintes condições são satisfeitas:

1) A adição de quaisquer dois vetores de V resulta em outro vetor em V (propriedade do fechamento).
2) O produto escalar de um elemento de {0, 1} e qualquer vetor de V resulta em outro vetor em V.
3) A lei distributiva é satisfeita, ou seja, se a_1 e a_2 são escalares de {0, 1} e $\mathbf{v_1}$ e $\mathbf{v_2}$ são vetores de V, então

$$\begin{aligned} a_1 \cdot (\mathbf{v_1} \oplus \mathbf{v_2}) &= (a_1 \cdot \mathbf{v_1}) \oplus (a_1 \cdot \mathbf{v_2}) \\ (a_1 \oplus a_2) \cdot \mathbf{v_1} &= (a_1 \cdot \mathbf{v_1}) \oplus (a_2 \cdot \mathbf{v_1}) \end{aligned} . \tag{2.6}$$

4) A lei associativa é satisfeita, ou seja, se a_1 e a_2 são escalares de {0, 1} e $\mathbf{v_1}$ é um vetor de V, então

$$(a_1 \cdot a_2) \cdot \mathbf{v_1} = a_1 \cdot (a_2 \cdot \mathbf{v_1}) . \tag{2.7}$$

EXEMPLO 2.4

Encontrar o espaço vetorial V composto pelo maior número possível de vetores com seis elementos de {0; 1}.

Solução:

O espaço vetorial V_6 é o conjunto de todos os vetores binários com seis elementos (n = 6) apresentados na Tabela 2.2.

Tabela 2.2 - Espaço vetorial V_6.

V_6							
000000	000001	000010	000011	000100	000101	000110	000111
001000	001001	001010	001011	001100	001101	001110	001111
010000	010001	010010	010011	010100	010101	010110	010111
011000	011001	011010	011011	011100	011101	011110	011111
100000	100001	100010	100011	100100	100101	100110	100111
101000	101001	101010	101011	101100	101101	101110	101111
110000	110001	110010	110011	110100	110101	110110	110111
111000	111001	111010	111011	111100	111101	111110	111111

* * *

Um subconjunto S de um espaço vetorial V é chamado de *subespaço vetorial de V* se as quatro condições definidas acima são verificadas. Entretanto, como S é um subconjunto de V, é suficiente que as duas primeiras condições sejam satisfeitas para a identificação de um subespaço em V, isto é:

1) A adição de quaisquer dois vetores de S resulta em outro vetor em S (propriedade do fechamento).
2) O produto escalar de um elemento de {0, 1} e qualquer vetor de S resulta em outro vetor em S. Ou simplesmente: o vetor nulo, ou vetor todo zero, pertence a S.

EXEMPLO 2.5

A partir das propriedades do subespaço vetorial identificar dois subespaços vetoriais de V_6, apresentado na Tabela 2.3, que contenha:

1) 4 vetores
2) 8 vetores

Solução:

Tabela 2.3 - Subespaços de V_6 com 4 e 8 vetores.

S com 4 vetores
000000
101011
110110
011101

S com 8 vetores	
000000	011001
110101	101011
101100	110010
011110	000111

Importante: Os subespaços encontrados acima não são únicos. É possível encontrar, em V_6, outros subespaços contendo 4 e 8 vetores.

* * *

2.3. CÓDIGOS DE BLOCO LINEARES [1][2][3][4][5][6]

Um *código de bloco linear binário* é um subespaço vetorial com 2^k vetores do espaço vetorial constituído de todos os 2^n vetores com n elementos de $\{0, 1\}$. Este conceito está ilustrado na Figura 2.3. Consequentemente, considerando as duas condições necessárias para caracterizar um subespaço vetorial aplicadas aos códigos de bloco lineares, conclui-se que:

1) A soma de duas palavras códigos quaisquer resulta em outra palavra código, e;
2) O vetor nulo ou vetor todo zero é também uma palavra código.

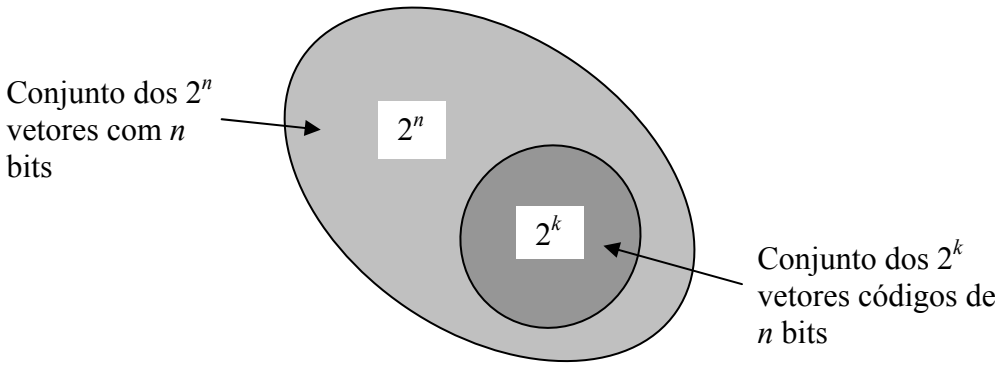

Figura 2.3 - Representação dos códigos de blocos lineares como um subespaço vetorial de um espaço vetorial V_n.

Nota-se que o subespaço vetorial constitui o conjunto dos vetores códigos ou vetores válidos. Portanto, qualquer vetor de n bits que não pertença ao subespaço vetorial está no espaço vetorial, porém, é um *vetor não válido*. Uma estratégia de detecção de erros consiste em verificar se o vetor recebido é um vetor válido ou não válido, i.e., se ele pertence ou não ao subespaço vetorial. Uma vez constatado que o vetor recebido é um vetor não válido, uma estratégia de correção de erros consiste na identificação de qual é o vetor válido que apresenta a menor distância de Hamming em relação ao vetor recebido e elegê-lo como sendo o vetor transmitido.

Nos códigos de bloco lineares onde o valor de k é baixo esta tarefa é simples. Entretanto, quando k apresenta valores relativamente altos, esta tarefa pode tornar-se impraticável, conforme mostrado no Exemplo 2.6.

EXEMPLO 2.6

Admitindo a existência de um código de bloco linear (255, 130), pede-se:

a) Determinar a quantidade de vetores códigos binários existentes neste código.
b) Determinar a quantidade de vetores não possíveis.
c) Descrever uma estratégia de correção de erros, admitindo que o vetor recebido é um vetor não válido.

Solução

a) A quantidade de vetores códigos é

$$2^k = 2^{130} \cong 1{,}36 \times 10^{39} \text{ vetores válidos.}$$

b) Como o vetor é não válido, então ele é um dos $2^n - 2^k$ vetores não válidos, ou um entre

$$2^{255} - 2^{130} \cong 2^{255} \cong 5{,}7896 \times 10^{76} \text{ vetores não válidos.}$$

c) Uma estratégia de correção de erros é identificar entre os $1{,}36 \times 10^{39}$ vetores válidos qual é o vetor que apresenta a menor distância de Hamming em relação ao vetor não válido recebido!

* * *

Nota-se que um subespaço vetorial é um conjunto de vetores *linearmente dependentes* (LD) devido à propriedade do fechamento, i.e., qualquer vetor pode ser obtido pela soma de outros dois vetores do subespaço. Uma vez que um subespaço vetorial binário contém 2^k vetores, então deve existir um ou mais subconjuntos com k vetores ditos *linearmente independentes* (LI) cujas combinações lineares produzem todos os outros vetores do subespaço. Esses k vetores linearmente independentes são chamados de *base do subspaço*. Os conceitos de espaço vetorial, subespaço vetorial e base do subespaço estão apresentados, em termos de conjunto, na Figura 2.4.

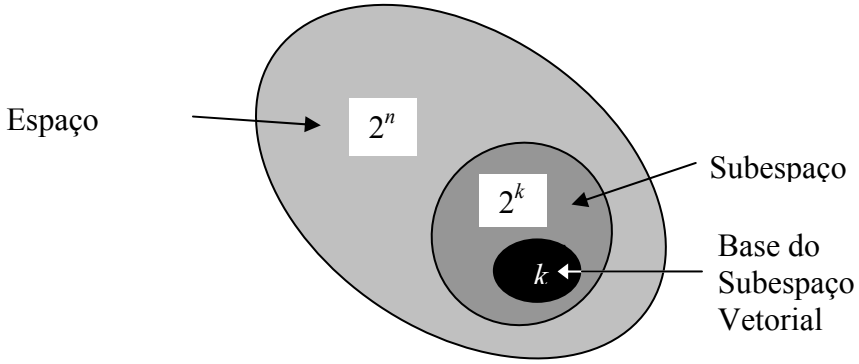

Figura 2.4. Representação de espaço vetorial, subespaço vetorial e base do subespaço vetorial.

EXEMPLO 2.7

A partir do subespaço vetorial com 8 vetores, apresentado na Tabela 2.3, identificar uma base capaz de gerar todos os outros vetores deste subespaço, através de combinações lineares dos vetores desta base.

Solução

Como o objetivo é formar uma base para a geração dos de um subespaço com $2^k = 8$ vetores LD o número necessário de vetores na base será $k = 3$ vetores LI. Escolhendo-se arbitrariamente 3 vetores LI entre os 8 vetores da Tabela 2.3, pode-se obter:

$$\mathbf{b_0} = 110101$$
$$\mathbf{b_1} = 101100$$
$$\mathbf{b_2} = 011110$$

Prova: Com a combinação linear dos vetores encontrados é possível encontrar todos os outros vetores do subespaço, representados a seguir pelos vetores $\mathbf{v}_j$, obtidos a partir da operação

$$\mathbf{v}_j = u_0 \cdot \mathbf{b}_0 + u_1 \cdot \mathbf{b}_1 + \ldots + u_{k-1} \cdot \mathbf{b}_{k-1}, \tag{2.8}$$

onde u_i é um elemento de $\{0, 1\}$ para $i = 1, 2, \ldots, (k-1)$ e $j = 0, 1, \ldots, (2^k - 1)$.

$\mathbf{v}_0 = 0 \cdot \mathbf{b}_0 + 0 \cdot \mathbf{b}_1 + 0 \cdot \mathbf{b}_2 = 000000$
$\mathbf{v}_1 = 1 \cdot \mathbf{b}_0 + 0 \cdot \mathbf{b}_1 + 0 \cdot \mathbf{b}_2 = 110101$ ← Vetor da base
$\mathbf{v}_2 = 0 \cdot \mathbf{b}_0 + 1 \cdot \mathbf{b}_1 + 0 \cdot \mathbf{b}_2 = 101100$ ← Vetor da base
$\mathbf{v}_3 = 0 \cdot \mathbf{b}_0 + 0 \cdot \mathbf{b}_1 + 1 \cdot \mathbf{b}_2 = 011110$ ← Vetor da base
$\mathbf{v}_4 = 1 \cdot \mathbf{b}_0 + 1 \cdot \mathbf{b}_1 + 0 \cdot \mathbf{b}_2 = 011001$
$\mathbf{v}_5 = 1 \cdot \mathbf{b}_0 + 0 \cdot \mathbf{b}_1 + 1 \cdot \mathbf{b}_2 = 101011$
$\mathbf{v}_6 = 0 \cdot \mathbf{b}_0 + 1 \cdot \mathbf{b}_1 + 1 \cdot \mathbf{b}_2 = 110010$
$\mathbf{v}_7 = 1 \cdot \mathbf{b}_0 + 1 \cdot \mathbf{b}_1 + 1 \cdot \mathbf{b}_2 = 000111$

* * *

2.3.1. MATRIZ GERADORA

Uma matriz geradora, **G**, é aquela que permite obter os vetores códigos, $\mathbf{c}_j$, correspondentes às mensagens, $\mathbf{m}_i$, a partir do produto interno determinado por

$$\mathbf{c}_j = \mathbf{m}_j \cdot \mathbf{G}, \tag{2.9}$$

Evidentemente, a matriz geradora, **G,** é uma consequência direta de uma base do subespaço vetorial. Ela é uma matriz de dimensões $k \times n$ que consiste do arranjo formado pelos vetores linearmente independentes, ou *vetores geradores*, que compõem uma base do subespaço, conforme apresentado em (2.10).

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} & g_{02} & \cdots & g_{0,\,n-1} \\ g_{10} & g_{11} & g_{12} & \cdots & g_{1,\,n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ g_{k-1,\,0} & g_{k-1,\,1} & g_{k-1,\,2} & \cdots & g_{k-1,\,n-1} \end{bmatrix} \tag{2.10}$$

Onde $\mathbf{g}_0$, $\mathbf{g}_1$, ... , $\mathbf{g}_{k-1}$, são os vetores geradores. Para uma conveniente simplificação da notação, a partir deste ponto, os vetores códigos $\mathbf{c}_j$ e $\mathbf{m}_j$ serão denotados simplesmente por **c** e **m**, respectivamente. Logo, substituindo (2.10) em (2.9), obtém-se

$$\mathbf{c} = \mathbf{m} \cdot \mathbf{G} = (m_0, m_1, \ldots, m_{k-1}) \bullet \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = m_0\mathbf{g}_0 + m_1\mathbf{g}_1 + \ldots + m_{k-1}\mathbf{g}_{k-1}. \tag{2.11}$$

Observa-se claramente a semelhança entre (2.8) e (2.11), o que significa que a combinação linear dos elementos dos vetores mensagem com as linhas da matriz geradora produzem vetores códigos que estão associados inequivocamente aos vetores mensagens que os produziu.

EXEMPLO 2.8

A partir da base do subespaço vetorial apresentado no Exemplo 2.7, pede-se:

a) Construir uma matriz geradora.
b) A partir da matriz geradora, construir uma tabela com os vetores mensagens e seus respectivos vetores códigos.

Solução:

a) A obtenção de uma matriz geradora a partir do Exemplo 2.7 é direta, pois ela nada mais é do que a base de um subespaço vetorial, logo, utilizando os mesmos vetores $\mathbf{b_0}$ = 110110, $\mathbf{b_1}$ = 101100 e $\mathbf{b_2}$ = 011101 para $\mathbf{g_0}$, $\mathbf{g_1}$ e $\mathbf{g_2}$, respectivamente, obtém-se

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{2.12}$$

b) Como a matriz geradora possui três linhas, os vetores resultantes de todas as combinações lineares serão $2^k = 2^3 = 8$, obtidos a partir de todos os vetores mensagens possíveis contendo de três bits. Consequentemente **G** é a matriz geradora de um código (6, 3), conforme mostrado a seguir.

Tabela 2.4. - Vetores códigos do códigos (6, 3) gerados a partir da matriz G em (2.12).

m	c = m.G	c
000	$\mathbf{c}_0$ = 0 (110101) + 0 (101100) + 0 (011110)	**000000**
100	$\mathbf{c}_1$ = 1 (110101) + 0 (101100) + 0 (011110)	**110101**
010	$\mathbf{c}_2$ = 0 (110101) + 1 (101100) + 0 (011110)	**101100**
110	$\mathbf{c}_3$ = 1 (110101) + 1 (101100) + 0 (011110)	**011001**
001	$\mathbf{c}_4$ = 0 (110101) + 0 (101100) + 1 (011110)	**011110**
101	$\mathbf{c}_5$ = 1 (110101) + 0 (101100) + 1 (011110)	**101011**
011	$\mathbf{c}_6$ = 0 (110101) + 1 (101100) + 1 (011110)	**110010**
111	$\mathbf{c}_7$ = 1 (110101) + 1 (101100) + 1 (011110)	**000111**

* * *

2.3.2. CODIFICAÇÃO SISTEMÁTICA E NÃO SISTEMÁTICA

Os vetores códigos apresentados na Tabela 2.4, gerados pela operação apresentada em (2.11), não apresentam explicitamente a mensagem que o gerou como sendo um segmento do próprio vetor código. Isso significa que neste tipo de codificação a mensagem passa a ser conhecida somente após o processo de decodificação. Essa forma de codificação é chamada de *codificação não sistemática*.

Uma característica desejável em um processo de codificação para um código de bloco linear é aquela que permite que o vetor código seja composto por dois segmentos: um segmento composto pelos $(n - k)$ bits de redundância que permitem a verificação da validade do vetor e outro segmento correspondente aos k bits da mensagem que gerou os bits de redundância. A disposição do segmento redundância e do segmento mensagem é uma questão de convenção. A convenção adotada neste texto está apresentada na Figura 2.5. A forma de codificação que permite a obtenção do vetor código nesse formato é chamada de *codificação sistemática*.

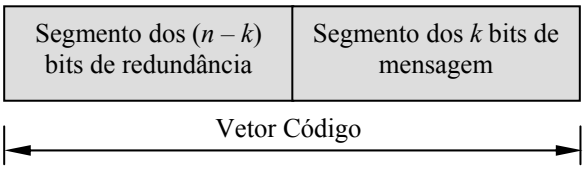

Figura 2.5 - Vetor código obtido por codificação sistemática.

Vetores códigos com a convenção mostrada na Figura 2.5 podem ser obtidos a partir de matrizes geradoras com um formato específico. Este formato, apresentado em (2.13), consiste de uma matriz geradora formada por duas outras matrizes: uma matriz de paridade com dimensões $k \times (n - k)$ e outra matriz identidade de dimensões $k \times k$. Desta forma, a matriz de paridade permite que o segmento paridade seja obtido pela soma linear dos bits da mensagem, enquanto a matriz identidade permite que o segmento mensagem seja replicado em seguida.

$$\mathbf{G} = \left[\mathbf{P}_{k\times(n-k)} \middle| \mathbf{I}_{k\times k}\right] = \left[\begin{array}{cccc|cccc} p_{00} & p_{01} & \cdots & p_{0,\,n-k-1} & 1 & 0 & \cdots & 0 \\ p_{10} & p_{11} & \cdots & p_{1,\,n-k-1} & 0 & 1 & & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ p_{k-1,\,0} & p_{k-1,\,1} & \cdots & p_{k-1,\,n-k-1} & 0 & 0 & \cdots & 1 \end{array}\right] = \left[\begin{array}{c} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{array}\right] \qquad (2.13)$$

Uma vez que uma matriz geradora é um arranjo de vetores linearmente independentes, uma matriz geradora de um código de bloco linear na forma sistemática pode ser obtida pela conveniente combinação linear dos vetores geradores e/ou permutação de colunas ou linhas da matriz geradora na forma não sistemática para a obtenção de outro arranjo de novos vetores geradores, linearmente independentes, no formato desejado. Essa operação é mostrada no Exemplo 2.9.

EXEMPLO 2.9

A partir da matriz geradora do código (6, 3) apresentada pela Equação (2.12), pede-se

a) Obter uma matriz geradora na forma sistemática, no formato apresentado em (2.13).
b) Construir uma tabela com os vetores mensagens e seus respectivos vetores códigos.

Solução

a) De (2.12),

$$\mathbf{G} = \begin{bmatrix} \mathbf{g_0} \\ \mathbf{g_1} \\ \mathbf{g_2} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

A matriz na forma sistemática, **G'** correspondente à matriz **G**, é obtida a partir das seguintes operações a partir da matriz **G**

$$\mathbf{g_0'} = \mathbf{g_1} = 101100$$
$$\mathbf{g_1'} = \mathbf{g_1} + \mathbf{g_2} = 110010$$
$$\mathbf{g_2'} = \mathbf{g_0} + \mathbf{g_1} = 011001$$

$$\mathbf{G'} = \begin{bmatrix} \mathbf{g_0'} \\ \mathbf{g_1'} \\ \mathbf{g_2'} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad (2.14)$$

b) Repetindo a operação apresentada em (2.11) para a matriz **G'** em (2.14) obtém-se os vetores códigos apresentados na Tabela 2.5.

Tabela 2.5 - Vetores códigos do códigos (5, 3) gerados a partir da matriz G' (2.14).

m	c = m.G'	c
000	$\mathbf{c}_0$ = 0 (101100) + 0 (110010) + 0 (011001)	**000000**
100	$\mathbf{c}_1$ = 1 (101100) + 0 (110010) + 0 (011001)	**101100**
010	$\mathbf{c}_2$ = 0 (101100) + 1 (110010) + 0 (011001)	**110010**
110	$\mathbf{c}_3$ = 1 (101100) + 1 (110010) + 0 (011001)	**011110**
001	$\mathbf{c}_4$ = 0 (101100) + 0 (110010) + 1 (011001)	**011001**
101	$\mathbf{c}_5$ = 1 (101100) + 0 (110010) + 1 (011001)	**110101**
011	$\mathbf{c}_6$ = 0 (101100) + 1 (110010) + 1 (011001)	**101011**
111	$\mathbf{c}_7$ = 1 (101100) + 1 (110010) + 1 (011001)	**000111**

Observe que tanto a matriz **G** quanto a matriz **G'** geram o mesmo subespaço vetorial. Entretanto, para cada uma das mensagens os vetores códigos gerados são diferentes em cada caso.

* * *

2.3.3. MATRIZ VERIFICADORA DE PARIDADE

Conforme apresentado na Figura 2.2, o vetor recebido, **r**, pode ser entendido como um vetor código que, ao ser transmitido através de um canal de comunicação, pode ter sofrido uma alteração, consequência da adição de um padrão de erro. Portanto, uma tarefa do decodificador é verificar se o vetor recebido é ou não um vetor código ou vetor válido. Mais uma vez, uma abordagem simplista para a realização desta tarefa seria a comparação do vetor recebido com todos os vetores códigos. Entretanto, para valores de k da ordem de algumas dezenas, esta abordagem pode tornar-se árdua, conforme mostrado no Exemplo 2.10. Uma forma mais simples para a verificação da validade ou não de um vetor recebido utiliza uma propriedade dos subespaços vetoriais, que pode ser definida da seguinte forma.

Se um subespaço vetorial, S_1 , pertence a um espaço vetorial, V_n , composto por todos os vetores de comprimento n, então deve existir um subespaço vetorial S_2 , que é o espaço nulo ou o espaço dual de S_1, e que pode ser representado por uma matriz composta por vetores bases linearmente independentes.

No estudo de códigos de bloco lineares a matriz geradora do subespaço nulo relativo ao subespaço gerado por **G** é chamada de matriz verificadora de paridade, cuja notação é **H**, e tem dimensões $(n - k) \times n$, ou seja,

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{n-k-1} \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} & \cdots & h_{0,\,n-1} \\ h_{10} & h_{11} & h_{12} & \cdots & h_{1,\,n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ h_{n-k-1,\,0} & h_{n-k-1,\,1} & h_{n-k-1,\,2} & \cdots & h_{n-k-1,\,n-1} \end{bmatrix}. \tag{2.15}$$

Se um subespaço gerado por **H** é dual ao subespaço gerado por **G** então os vetores de **G** são ortogonais aos vetores de **H,** ou seja,

$$\mathbf{G} \cdot \mathbf{H}^T = 0 . \tag{2.16}$$

Pode-se verificar sem dificuldades que uma consequência direta de (2.16) é que a condição de ortogonalidade de qualquer vetor código, **c,** gerado por **G** em relação ao espaço nulo gerado por **H** é verdadeira, i.e.,

$$\mathbf{c} \cdot \mathbf{H}^T = 0 . \tag{2.17}$$

Quando a matriz **G** está na forma sistemática, conforme apresentada em (2.13), ou seja,

$$\mathbf{G} = \left[\mathbf{P}_{k\times(n-k)} \middle| \mathbf{I}_{k\times k} \right] ,$$

a obtenção da matriz **H** é direta, conforme mostrado a seguir.

$$\mathbf{H} = \left[\mathbf{I}_{(n-k)\times(n-k)} \middle| \mathbf{P}^T\right] = \left[\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & p_{00} & p_{10} & \cdots & p_{k-1,0} \\ 0 & 1 & & 0 & p_{01} & p_{11} & \cdots & p_{k-1,1} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \cdots & p_{k-1,n-k-1} \end{array}\right] \tag{2.18}$$

EXEMPLO 2.10

A partir da matriz geradora para o código (6, 3) apresentada em (2.14), pede-se:

a) Obter a matriz verificadora de paridade **H**.
b) Verificar a condição de ortogonalidade apresentada por (2.17) para o vetor código correspondente ao vetor mensagem **m** = 101.

Solução:

a) A partir da matriz geradora na forma sistemática

$$\mathbf{G}' = \left[\mathbf{P}_{k\times(n-k)} \middle| \mathbf{I}_{k\times k}\right] = \left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

obtém-se a matriz **H** na forma

$$\mathbf{H} = \left[\mathbf{I}_{(n-k)\times(n-k)} \middle| \mathbf{P}^T\right] = \left[\begin{array}{cccccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right]. \tag{2.19}$$

b) O vetor código, **c,** correspondente ao vetor mensagem **m** = 101 pode ser obtido conforme mostrado em (2.11), ou seja:

$$\mathbf{c} = \mathbf{m}\cdot\mathbf{G}' = (101)\left[\begin{array}{cccccc} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right] = 1(101100) + 0(110010) + 1(011001) = 110101$$

A condição de ortogonalidade pode ser verificada a partir do resultado do produto interno entre o vetor código, **c**, e a matriz verificadora de paridade transposta $\mathbf{H}^T$, conforme mostrado a seguir.

$$\mathbf{c}\cdot\mathbf{H}^T = (110101)\left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{array}\right] = 1(100) + 1(010) + 0(001) + 1(101) + 0(110) + 1(011) = 000. \tag{2.20}$$

* * *

2.3.4. DISTÂNCIA MÍNIMA DE UM CÓDIGO DE BLOCO LINEAR

Considere o conjunto de distâncias entre todos os pares de vetores código em um espaço V_n. O menor membro do conjunto é a *distância mínima do código* e é denotado por $d_{\min}$.

Mais uma vez a propriedade dos códigos lineares discutida anteriormente permite afirmar que se $\mathbf{c_1}$ e $\mathbf{c_2}$ são vetores código, então o vetor $\mathbf{c_3}$ **o**btido pela operação $\mathbf{c_1} \oplus \mathbf{c_2}$ é também um vetor código. Assim a distância de Hamming entre dois vetores códigos é determinada como sendo

$$d(\mathbf{c}_1, \mathbf{c}_2) = w(\mathbf{c}_1 \oplus \mathbf{c}_2) = w(\mathbf{c}_3) \tag{2.21}$$

Portanto, não há necessidade de examinarmos as distâncias entre todas as combinações possíveis entre pares de palavras-código, basta que se verifique o peso de cada palavra código, com exceção da palavra toda zero. O menor peso encontrado corresponde à menor distância mínima do código.

As propriedades dos códigos de blocos permitem ainda determinarmos a distância mínima do código através da inspeção da matriz verificadora de paridade. Neste caso, a distância mínima do código será igual ao menor número de colunas da matriz verificadora de paridade, que quando somadas resultam em uma coluna toda zero. Este procedimento é particularmente útil quando o código possui um número muito grande de palavras código para serem inspecionadas, sem auxílio computacional.

Exemplo 2.11

Determine a distância mínima do código (6, 3), definida pela matriz **G** (2.14), repetida abaixo por conveniência.

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

Solução:

Uma vez que este código possui poucas palavras códigos, uma solução é listá-las por meio da operação **c** = **m.G**. O resultado dessa operação para todas as possíveis mensagens com 3 bits está mostrado na tabela a seguir.

Tabela 2.6 - Vetores códigos do códigos (6, 3) gerados a partir da matriz **G** (2.14).

m	c
000	000000
100	101100
010	110010
110	011110
001	011001
101	110101
011	101011
111	000111

Pode-se verificar, inspecionando-se a tabela acima, que as palavras de menor peso são as palavras com peso 3. Logo a distância mínima é igual a 3.

Ou alternativamente, inspecionando-se a matriz verificadora de paridade, apresentada abaixo, é fácil verificar que o menor número de colunas que quando somadas resulta em uma coluna toda zero é 3, por exemplo, $1^{a} + 2^{a} + 5^{a}$ ou $2^{a} + 3^{a} + 6^{a}$ ou $1^{a} + 3^{a} + 4^{a}$.

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{2.22}$$

* * *

É muito comum um código de bloco linear (n, k) com distância mínima d_{min} ser representado pela notação (n, k, d_{min}). Assim, o código (6, 3) do Exemplo 2.11 é um código (6, 3, 3).

2.3.5. CAPACIDADE DE CORREÇÃO E DE DETECÇÃO DE ERROS DE UM CÓDIGO DE BLOCO LINEAR

No receptor, o decodificador tem por função estimar o vetor código recebido em função do vetor código transmitido. Em um canal de transmissão *AWGN* (Ruído Branco Gaussiano Aditivo), o ruído afeta os símbolos transmitidos aleatoriamente, segundo uma distribuição estatística *normal* ou *gaussiana*. Assim, padrões de erros com menos bits errados tem maior probabilidade de ocorrer do que padrões de erros com mais bits errados.

Consequentemente, em um canal BSC (Canal Simétrico Binário), dado um vetor recebido **r**, a melhor estimativa é feita admitindo-se que o vetor código transmitido é aquele que está mais próximo de **r**, sob o ponto de vista da distância de Hamming. Se dois vetores códigos tiverem a mesma distância de Hamming do vetor **r** recebido, a escolha pode ou não ser arbitrária dependendo do tipo de decisor utilizado (*hard decision* ou *soft decision*).

A Figura 2.6 apresenta dois vetores $\mathbf{c_1}$ e $\mathbf{c_2}$ unidos por uma linha calibrada em distância de Hamming. Cada ponto preto representa um vetor recebido **r**. Na parte (a) da figura, o vetor recebido $\mathbf{r}_1$ dista 1 bit de $\mathbf{c_1}$ e 4 bits de $\mathbf{c_2}$. De acordo com a estratégia de máxima probabilidade, o decodificador selecionará o vetor $\mathbf{c_1}$ como aquele que foi transmitido. Na parte (b) da figura, o vetor recebido $\mathbf{r}_2$ dista 2 bits de $\mathbf{c_1}$ e 3 bits de $\mathbf{c_2}$. Mais uma vez o decodificador selecionará o vetor $\mathbf{c_1}$ como sendo o vetor recebido. Finalmente na parte (c) da figura, o vetor recebido $\mathbf{r}_3$ dista 3 bit de $\mathbf{c_1}$ e 2 bits de $\mathbf{c_2}$. Desta vez o decodificador selecionará o vetor $\mathbf{c_2}$ como sendo o vetor recebido.

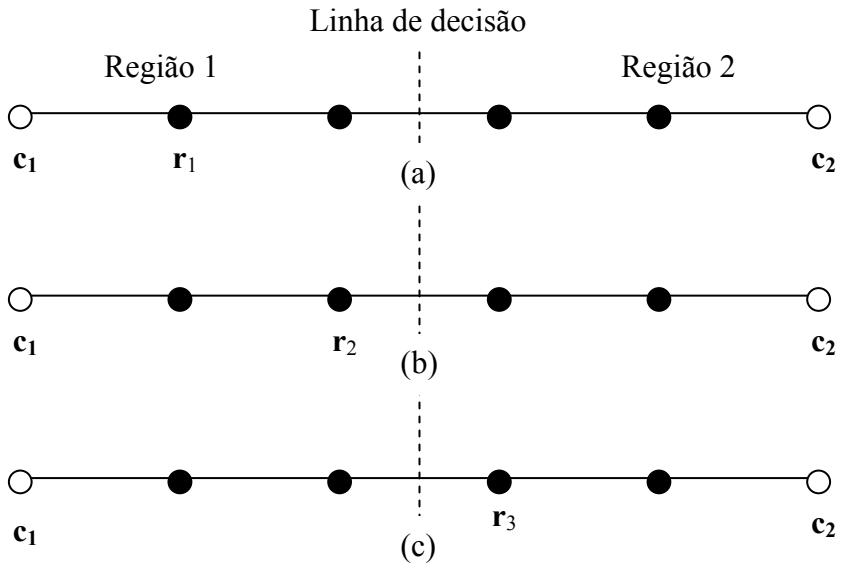

Figura 2.6 – Capacidade de detecção e correção de erro. (a) Vetor recebido $\mathbf{r}_1$. (b) Vetor recebido $\mathbf{r}_2$. (c) Vetor recebido $\mathbf{r}_3$.

Consequentemente, a capacidade de correção de erro, de um código de bloco linear, t, é definida como o número máximo de erros garantidamente corrigíveis, por palavra código, determinado por

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor \tag{2.23}$$

onde $\lfloor x \rfloor$ significa o maior inteiro que não excede o valor de x. Assim, um código que corrige todas as sequências de t erros, pode também corrigir certas sequências de t +1 erros.

Se o código for usado com a finalidade exclusiva de detectar erros ao invés de corrigir erros, a capacidade de detecção de erros do código é determinada por

$$e = d_{\min} - 1, \tag{2.24}$$

onde e é o número de erros detectados. Entretanto, é possível utilizar um código de bloco para corrigir τ erros e detectar ε erros simultaneamente desde que $\tau < t$ e $\varepsilon < e$, e a seguinte condição seja satisfeita:

$$d_{\min} = \tau + \varepsilon + 1 . \tag{2.25}$$

Assim, um código com $d_{\min}$ = 7, por exemplo, pode corrigir todos os padrões de 3 erros (t = 3) ou detectar todas as combinações possíveis de até 6 erros (e = 6). Entretanto, se ele for usado para corrigir até 2 erros (τ = 2), ele pode detectar simultaneamente padrões de até 4 erros (ε = 4), pois neste caso a condição apresentada por (2.25) é satisfeita.

2.3.6. SÍNDROME DE ERRO

Seja $\mathbf{r} = r_1, r_2, \ldots, r_n$ um vetor recebido (i.e. uma das 2^n palavras de n bits do espaço vetorial V_n) resultante da transmissão de um vetor código $\mathbf{c} = c_1, c_2, \ldots, c_n$ (i.e. uma das 2^k palavras códigos de n bits) através de um canal com ruído. Logo,

$$\mathbf{r} = \mathbf{c} + \mathbf{e}\,. \tag{2.26}$$

onde $\mathbf{e} = e_1, e_2, \ldots, e_n$ é um vetor erro ou padrão de erro introduzido pelo canal. *Síndrome de erro* é um vetor com $n - k$ bits definido pela operação

$$\mathbf{S} = \mathbf{r} \cdot \mathbf{H}^T \tag{2.27}$$

Combinando (2.26) e (2.27) tem-se:

$$\mathbf{S} = (\mathbf{c} + \mathbf{e}) \cdot \mathbf{H}^T = \mathbf{c} \cdot \mathbf{H}^T + \mathbf{e} \cdot \mathbf{H}^T \tag{2.28}$$

mas,

$$\mathbf{c} \cdot \mathbf{H}^T = 0\,. \tag{2.29}$$

Consequentemente,

$$\mathbf{S} = \mathbf{e} \cdot \mathbf{H}^T \tag{2.30}$$

A equação acima mostra que a síndrome **S** está associada a um padrão de erro. Esta é uma importante propriedade, fundamental para o processo de decodificação, ou seja, cada padrão de erro corrigível deve estar associado a uma síndrome específica.

É importante notar que para isso ocorra, duas propriedades da matriz verificadora de paridade são necessárias:

*Nenhuma coluna da matriz **H** pode ser toda zero, caso contrário, um erro na posição correspondente à linha toda zero seria indetectável;*

*Todas as colunas de **H** devem ser únicas. Se duas colunas de **H** forem iguais, erros nas posições correspondentes a essas linhas podem ser indistinguíveis.*

Um código de bloco linear (n, k) com capacidade de correção de t erros é capaz de corrigir um total de 2^{n-k} padrões de erros.

Os códigos que corrigem exclusivamente todos os padrões de t erros ou menos e nenhum padrão maior que t erros, são denominados *códigos perfeitos*, isto é, o número de síndromes deve ser igual ao número exato de padrões com até t erros e nenhum padrão de erro contendo um número maior do que t erros. Uma vez que todo código de bloco é capaz de corrigir 2^{n-k} padrões de erros, um código é perfeito quando a igualdade apresentada a seguir é satisfeita.

$$2^{n-k} = \sum_{i=0}^{t} \binom{n}{i} \tag{2.31}$$

EXEMPLO 2.12

Considere o código de bloco linear (6, 3, 3) gerado por (2.14). Pede-se:

a) Determinar a capacidade de correção de erros do código.
b) Listar todos os padrões de erros corrigíveis dentro da capacidade de correção de erros do código.
c) Listar todas as síndromes de erros associadas aos padrões de erros corrigíveis dentro da capacidade de correção de erros do código.
d) Verificar se este código é um código perfeito.

Solução:

a) A capacidade de correção de erros do código.

Como a distância mínima deste código é $d_{min} = 3$, então,

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor = \left\lfloor \frac{3-1}{2} \right\rfloor \quad \Rightarrow \quad t = 1$$

b) Padrões de erros corrigíveis dentro da capacidade de correção de erros do código

Como a capacidade de correção de erro é $t = 1$, então este código é capaz de corrigir todos os padrões com 1 erro, ou seja,

e ($t = 1$)
100000
010000
001000
000100
000010
000001

c) Síndromes de erros associadas aos padrões de erros corrigíveis dentro da capacidade de correção de erros do código.

Da matriz **H** (2.22), obtém-se $\mathbf{H}^T$, ou seja,

$$\mathbf{H}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}. \qquad (2.32)$$

De (2.30),

$$\mathbf{S} = \mathbf{e} \cdot \mathbf{H}^T \ .$$

Para todos os valores de **e** ($t = 1$) obtém-se:

Tabela 2.7 - Padrões de erros corrigíveis e suas respectivas síndromes para o código de bloco linear (6, 3, 3) gerado por (2.14).

e ($t = 1$)	S
100000	**100**
010000	**010**
001000	**001**
000100	**101**
000010	**110**
000001	**011**

d) Verificação se este código é um código perfeito.

Um código perfeito deve satisfazer

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

$$2^{n-k} = 2^{6-3} = 8$$

$$\sum_{i=0}^{t} \binom{n}{i} = \binom{6}{0} + \binom{6}{1} = 1 + 6 = 7$$

Logo, o código de bloco linear (6, 3, 3) não é um código perfeito. De fato, o número de síndromes possíveis são todas as síndromes não nulas mais a síndrome nula, que totalizam 7 síndromes. Por inspeção à Tabela 2.7, verifica-se a ausência da nula e da síndrome **S** = 111. A ausência da síndrome nula deve-se ao fato que ela corresponde ao padrão de todo zero, ou seja, vetor recebido sem erro, enquanto que a síndrome 111 só ocorrerá se o padrão de erro tiver 2 ou mais erros, o que está fora da capacidade de correção deste código.

2.3.7. CORREÇÃO DE ERROS PELA SÍNDROME

Conforme mostrado no Exemplo 2.12, deve existir uma correspondência exclusiva entre um padrão de erro e uma síndrome de erro. Isso abre a possibilidade de não só podermos detectar erros, mas também corrigi-los. A correção de erros pode ser feita de diversas formas. A seguir será apresentada a correção de erros por meio da síndrome de erros. Basicamente, este processo de correção é feito a partir da identificação do padrão de erro mais provável por meio do cálculo da síndrome de erros. Uma vez conhecido o padrão de erro é possível fazer a correção de erro somando-se o vetor **r** recebido com o padrão de erro, **e**, associado a ele, pois em (2.26),

$$\mathbf{r} = \mathbf{c} + \mathbf{e},$$

então,

$$\mathbf{c}' = \mathbf{r} + \mathbf{e}. \qquad (2.33)$$

Onde **c'** é a melhor estimativa do vetor código que foi transmitido pelo canal ruidoso. Este procedimento pode ser resumido de acordo com os seguintes passos:

1. A partir da capacidade de correção de erros do código, determina-se a síndrome para todos os padrões de erros corrigíveis, por meio (2.30);
2. Calcula-se a síndrome de **r** usando (2.27);
3. Localiza-se o padrão de erro correspondente à síndrome calculada.
4. O vetor código será aquele determinado por (2.33).

Observação: O erro só será corrigido se o padrão de erro correspondente à síndrome de vetor recebido for igual ao padrão de erro introduzido pelo canal, i.e., o padrão de erro introduzido pelo canal deve ser um padrão de erro corrigível.

EXEMPLO 2.13

Suponha que o vetor **c** = 101011 do código (6, 3, 3), gerado por (2.14), tenha sido transmitido e corrompido por ruído no canal, de modo que na recepção foi detectado o vetor **r** = 101010. Corrija o erro introduzido pelo canal a partir da associação da síndrome com o padrão de erro mais provável.

Solução:

A síndrome de erros para o vetor recebido é determinada por meio de (2.33), ou seja:

$$\mathbf{S} = \mathbf{r} \cdot \mathbf{H}^T = (101010)\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = 1(100) + 0(010) + 1(001) + 0(101) + 1(110) + 0(011)$$

$$\mathbf{S} = 100 + 001 + 110 \qquad \Rightarrow \qquad \mathbf{S} = 011$$

Em canais AWGN os padrões de erros mais prováveis são aqueles com menor número de erros. Além disso, o erro só será corrigido com certeza se o padrão de erro introduzido pelo canal for um padrão de erro corrigível pelo código.

A Tabela 2.7 obtida no Exemplo 2.12 associa os padrões de erros às suas respectivas síndromes. Esta tabela é reapresentada a seguir com as colunas invertidas e reordenadas, para permitir mais facilmente a identificação do padrão de erro a partir da síndrome obtida acima.

Tabela 2.8 - Síndromes e seus respectivos padrões para o código de bloco linear (6, 3, 3) gerado por (2.27).

S	**e** ($t = 1$)
000	**000000**
001	**001000**
010	**010000**
011	**000001**
100	**100000**
101	**000100**
110	**000010**

De acordo a Tabela 2.8 a síndrome calculada corresponde ao padrão de erro **e** = 000001. Logo, o vetor código mais provável de ter sido o vetor transmitido pode ser determinado por (2.39), i.e.:

$$\mathbf{c}' = \mathbf{r} + \mathbf{e} = 101010 + 000001$$
$$\mathbf{c}' = 101011$$

* * *

- **ARRANJO PADRÃO**

O *arranjo padrão* é um esquema de decodificação baseado em síndrome de erro. Apesar da sua importância didática, sua aplicabilidade fica restrita aos códigos de bloco com número reduzido de palavras códigos.

Conforme já apresentado, um código de bloco é um subespaço vetorial composto por 2^k palavras códigos. Entretanto, quando uma palavra código é transmitida por um canal ruidoso e é corrompida por ruído, ela pode se transformar em uma palavra não válida que pertence a um conjunto de 2^n palavras com n bits, que constitui o espaço vetorial V_n.

Considere as 2^k palavras códigos de um código de bloco linear $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{2^k}$.

Considere também os 2^{n-k} padrões de erros $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{2^{n-k}}$ associados às 2^{n-k} síndromes possíveis.

Um arranjo padrão é constituído por todas as palavras do espaço vetorial V_n, de acordo com os passos apresentados a seguir e ilustrados na Figura 2.7.

1. Um arranjo padrão é formado por 2^k subconjuntos, sendo que cada subconjunto é uma coluna do arranjo padrão. Consequentemente, um arranjo padrão possui 2^k colunas.
2. A primeira linha do arranjo padrão é composta por todas as 2^k palavras códigos, ou seja, a primeira linha do arranjo padrão é o subespaço vetorial das 2^k palavras códigos. Obrigatoriamente a palavra toda zero ou chamada de $\mathbf{c}_1 = \mathbf{e}_1 = 0$ é a palavra que ocupa a posição superior esquerda do arranjo.
3. A primeira coluna do arranjo padrão é formada por todos os 2^{n-k} padrões de erros, incluindo o vetor todo zero que ocupa a posição superior esquerda do arranjo. Consequentemente, um arranjo padrão possui 2^{n-k} linhas. A segunda linha do arranjo é formada pela soma de $\mathbf{e}_2$ com cada um dos vetores códigos na primeira linha. Este procedimento se repete até que a (n-k) é-sima linha complete o arranjo.

Cada linha do arranjo forma um subconjunto disjunto dos demais subconjuntos das outras linhas. Cada subconjunto é chamado de *coset* e o vetor do *coset* que pertence a primeira coluna é chamado de *lider do coset*.

O arranjo padrão permite a decodificação direta da palavra recebida pela identificação de um dos 2^k subconjunto, ou coluna, ao qual pertence a palavra recebida. Uma vez identificada esta coluna o vetor código decodificado é o vetor da primeira linha da coluna identificada.

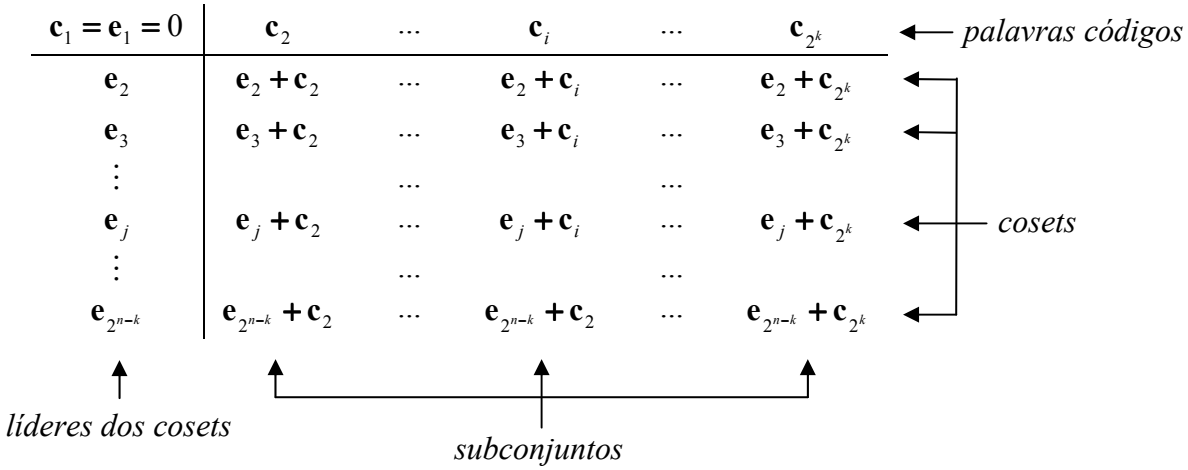

Figura 2.7 - Arranjo padrão.

O Exemplo 2.14 apresenta o uso do arranjo padrão para a correção de erros e também um comentário sobre o equívoco de decodificação que pode ocorrer quando o código usado não é um código perfeito.

EXEMPLO 2.14

Considere o código (6, 3, 3), gerado por (2.14). Pede-se:

a) Construir um arranjo padrão para este código.
b) Decodificar o vetor recebido 101010.
c) Admita que o vetor código 101100 tenha sido corrompido pelo padrão de erro 010100. Decodifique o vetor recebido.

Solução:

a) O Arranjo Padrão

De acordo com o Exemplo 2.13, este código é capaz de corrigir, garantidamente, todos os padrões de um erro. Como $n = 6$ então existem seis padrões de um erro que são aqueles listados na Tabela 2.7. Entretanto, para a montagem do arranjo padrão são necessários $2^{(n-k)} = 8$ padrões de erros para compor a primeira coluna do arranjo. Esses oito padrões de erros geram, evidentemente, oito síndromes de erro, incluindo o padrão de erro e = 000000, associado à síndrome **S** = 111, que ocupa a primeira célula da primeira coluna. Logo, Conforme apresentado na Tabela 2.8 a única síndrome não nula não associada aos padrões de um erro é a síndrome

S = 111. Isso significa que deve haver um ou mais padrões de duplo erro associado à síndrome **S** = 111, pois a capacidade de correção deste código é de um erro. De fato, pode-se verificar que existem três padrões de duplo erro capaz de gerar a síndrome **S** = 111, conforme mostrado a seguir, mas apenas uma deve ser escolhida para compor o arranjo padrão.

$$\mathbf{S} = \mathbf{e} \cdot \mathbf{H}^T = \left.\begin{matrix} (100001) \\ ou \\ (010100) \\ ou \\ (001010) \end{matrix}\right\} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \quad \Rightarrow \quad \mathbf{S} = 111$$

Uma vez escolhido o padrão de duplo erro para completar a primeira linha do arranjo padrão, que neste exemplo é o padrão de erro **e** = 100001, pode-se montar o arranjo padrão da seguinte forma:

1) 1ª Linha: todas as palavras códigos iniciando-se pela palavra toda zero (Tabela 2.6).

000000	**101100**	**110010**	**011110**	**011001**	**110101**	**101011**	**000111**

2) 1ª Coluna: os (*n* - *k*) padrões de erros corrigíveis (garantidamente ou não).

000000	**101100**	**110010**	**011110**	**011001**	**110101**	**101011**	**000111**
100000							
010000							
001000							
000100							
000010							
000001							
100001							

3) Demais células: cada célula é ocupada pelo vetor resultante da soma da palavra código do subconjunto da célula pelo seu líder do *coset*.

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

b) Decodificação do vetor 101010

Conforme mostrado abaixo, o vetor 101010 pertence ao *coset* cujo líder é o padrão de erro 000001 e ao subconjunto do vetor código 101011.

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

Deste modo, a decodificação pelo arranjo padrão pressupõe que o vetor código transmitido foi o vetor 101011 (vetor decodificado) que foi corrompido pelo padrão de erro 000001.

c) Decodificação do vetor **c** = 101100 corrompido por **e** = 010100

$$\mathbf{r} = \mathbf{c} + \mathbf{e} = 101100 + 010100 \quad \Rightarrow \quad \mathbf{r} = 111000$$

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

A decodificação pelo arranjo padrão pressupõe que o vetor código transmitido foi o vetor 011001 (vetor decodificado) que foi corrompido pelo padrão de erro 100001. Note que houve um equívoco na decodificação porque existem três padrões de duplo erro associado à síndrome **S** = 111 e o que foi escolhido para compor o arranjo padrão não foi o padrão de duplo erro introduzido pelo canal.

* * *

2.3.8. CODIFICADOR PARA CÓDIGOS DE BLOCO SISTEMÁTICOS SIMPLES

Quando os códigos de blocos são simples e curtos, a implementação de um circuito de codificação pode ser feita diretamente com o auxílio das equações que determinam as palavras códigos. Para um código de bloco linear sistemático, a palavra código pode ser escrita como

$$\mathbf{c} = (p_0, p_1, \ldots, p_{n-k-1}, m_0, m_1, \ldots, m_{k-1}). \tag{2.34}$$

Note que a obtenção dos bits de paridade a partir de (2.13) resume-se às seguintes operações:

$$\begin{aligned} p_0 &= m_0 \cdot p_{00} + m_1 \cdot p_{10} + \ldots + m_{k-1} \cdot p_{k-1,0} \\ p_1 &= m_0 \cdot p_{01} + m_1 \cdot p_{11} + \ldots + m_{k-1} \cdot p_{k-1,1} \\ &\vdots \\ p_{n-k-1} &= m_0 \cdot p_{0,n-k-1} + m_1 \cdot p_{1,n-k-1} + \ldots + m_{k-1} \cdot p_{k-1,n-k-1} \end{aligned} \tag{2.35}$$

Assim, com o conhecimento da matriz geradora na forma sistemática é possível particularizar o conjunto de equações apresentadas em (2.35) para a geração das palavras códigos por meio de circuitos lógicos combinacionais. Veja Exemplo 2.15.

EXEMPLO 2.15

Construa um codificador para o código (6, 3) representado pela sua matriz geradora reproduzida a seguir

$$\mathbf{G} = \left[\mathbf{P}_{k\times(n-k)} \middle| \mathbf{I}_{k\times k}\right] = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Solução:

De acordo com (2.41) as equações de paridade para a matriz geradora acima são:

$$\begin{aligned} p_0 &= m_0 + m_2 \\ p_1 &= m_1 + m_2 \\ p_2 &= m_0 + m_1 \end{aligned}$$

Logo, um circuito codificador pode ser implementado conforme apresentado a seguir:

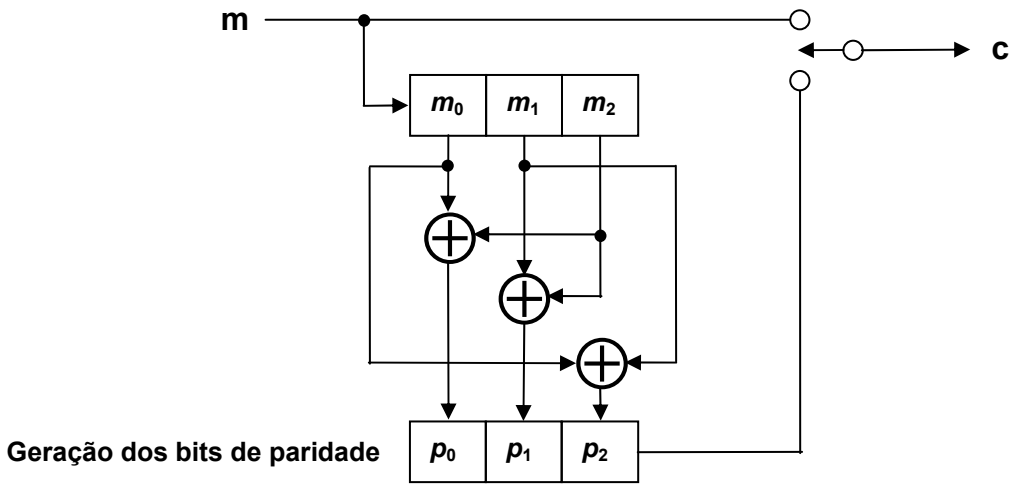

Figura 2.8 - Circuito de codificação para o código (6, 3).

* * *

2.3.9. DECODIFICADOR PARA CÓDIGOS DE BLOCO SISTEMÁTICOS SIMPLES

Quando os códigos de blocos são simples e curtos um decodificador pode ser implementado com um circuito simples a partir dos seguintes passos:

1. Calculo da síndrome;
2. Localização do padrão de erro;
3. Soma módulo-2 do padrão de erro com o vetor recebido.

O cálculo da síndrome pode ser feito considerando as expressões apresentadas a seguir.

$$\mathbf{S} = \mathbf{r} \cdot \mathbf{H}^T = (r_0, r_1, \ldots, r_{n-1}) \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \\ p_{00} & p_{01} & \cdots & p_{0,\,n-k-1} \\ p_{10} & p_{11} & \cdots & p_{1,\,n-k-1} \\ \vdots & \vdots & & \vdots \\ p_{k-1,\,0} & p_{k-1,\,1} & \cdots & p_{k-1,\,n-k-1} \end{bmatrix} = (s_0, s_1, \ldots, s_{n-k-1}) \tag{2.36}$$

$$\begin{aligned} s_0 &= r_0 + r_{n-k} \cdot p_{00} + r_{n-k+1} \cdot p_{10} + \ldots + r_{n-1} \cdot p_{k-1,0} \\ s_1 &= r_1 + r_{n-k} \cdot p_{01} + r_{n-k+1} \cdot p_{11} + \ldots + r_{n-1} \cdot p_{k-1,1} \\ &\vdots \\ s_{n-k-1} &= r_{n-k-1} + r_{n-k} \cdot p_{0,n-k-1} + r_{n-k+1} \cdot p_{1,n-k-1} + \ldots + r_{n-1} \cdot p_{k-1,n-k-1} \end{aligned} \tag{2.37}$$

Cada síndrome deve gerar um padrão de erro (corrigível) que deverá ser somado ao vetor recebido.

EXEMPLO 2.16

Construa um decodificador para o código (6, 3) representado pela sua matriz verificadora de paridade transposta reproduzida a seguir.

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

Solução:

De acordo com (2.37) as equações das síndromes a partir da matriz verificadora de paridade acima são:

$$s_0 = r_0 + r_3 + r_4$$
$$s_1 = r_1 + r_4 + r_5$$
$$s_2 = r_2 + r_3 + r_5$$

Seguindo os passos apresentados e utilizando as equações acima, pode-se desenhar o circuito de decodificação apresentado a seguir. Note que o circuito de decodificação pode ser simplificado por meio da exclusão das portas acinzentadas, responsáveis pelos bits c_0, c_1 e c_2, de paridade, uma vez que o resultado da decodificação resume-se aos bits de informação c_3, c_4 e c_5.

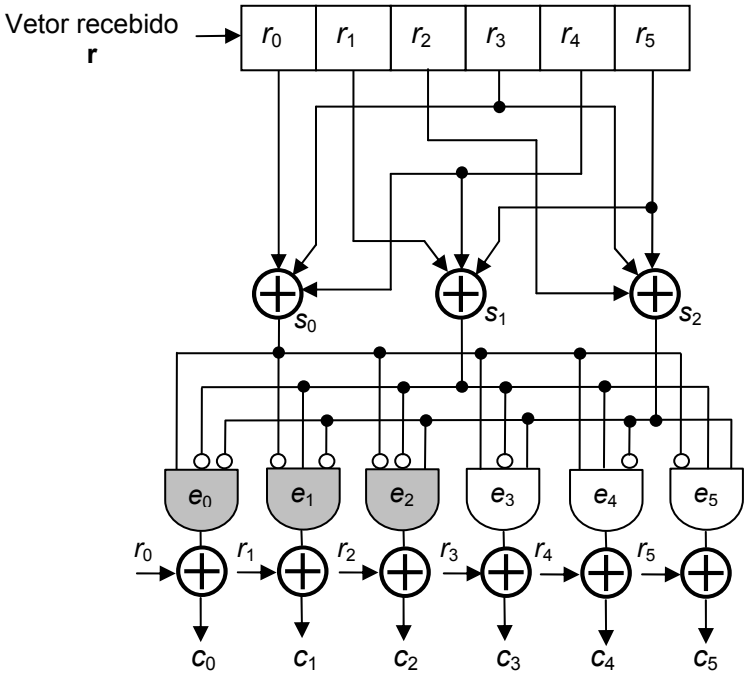

Figura 2.9 - Circuito de decodificação para o código (6, 3).

* * *

2.4. DESEMPENHO DOS CÓDIGOS DE BLOCO LINEARES [1]

Se um código de bloco capaz de corrigir t erros é utilizado exclusivamente para correção de erros em um canal binário simétrico (BSC), com probabilidade de transição (probabilidade de erro de bit) p, a probabilidade de erro de bit após a decodificação pode ser determinada aproximadamente por:

$$P_b \approx \frac{1}{n} \sum_{j=t+1}^{n} j \binom{n}{j} p^j (1-p)^{n-j} . \tag{2.38}$$

Para fins de comparação, o desempenho de alguns códigos de bloco lineares simples em um canal AWGN com modulação BPSK, com detecção coerente, pode ser obtido considerando-se que a

probabilidade de erro de símbolo em termos da relação entre a energia de símbolo codificado e a densidade espectral de ruído, E_c/N_0, pode ser determinada por

$$p = \frac{1}{2} erfc \sqrt{\frac{E_c}{N_0}} . \tag{2.39}$$

Por sua vez é necessário relacionar também E_c/N_0 com Eb/N_0, que é relação entre energia gasta com os bits de informação e a densidade espectral de ruído, ou seja,

$$\frac{E_c}{N_0} = \left(\frac{k}{n}\right)\frac{E_b}{N_0} = R_c \frac{E_b}{N_0} . \tag{2.40}$$

Substituindo (2.46) em (2.45) obtém-se:

$$p = \frac{1}{2} erfc \sqrt{R_c \frac{E_b}{N_0}} . \tag{2.41}$$

Com (2.41) e (2.38) a comparação do desempenho aproximado entre alguns códigos de blocos simples transmitidos sobre a modulação BPSK com a própria modulação BPSK não codificada pode ser obtida diretamente. A Figura 2.10 mostra o desempenho aproximado entre a modulação BPSK não codificada com as BPSKs com os códigos: *Hamming* (7, 4), *BCH* (15, 7), *Golay* (23, 12), *BCH* (63, 36) e *BCH* (127, 64).

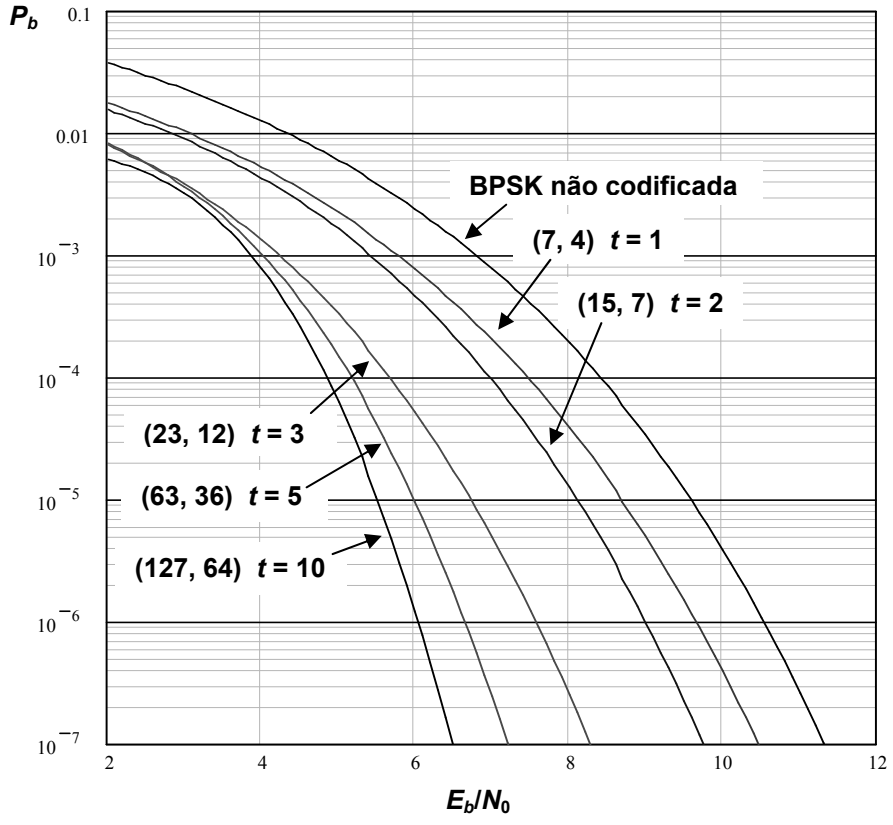

Figura 2.10 - Desempenho de alguns códigos de blocos lineares.

Note que os códigos, cujos desempenhos estão apresentados na Figura 2.10, possuem taxa de codificação entre 0,47 e 0,57. Procurou-se, desta forma, manter a taxa de codificação praticamente constante para que o desempenho dos códigos com diferentes comprimentos fossem comparados. Isso foi feito com o objetivo de evidenciar uma importante característica dos códigos de blocos: a capacidade dc correção de erros por bloco aumenta com o aumento do comprimento quando a taxa de codificação é mantida constante. Como consequência, aumentos significativos no desempenho podem ser obtidos.

2.5. CÓDIGOS CÍCLICOS [1][2][3][4][5][6]

2.5.1. INTRODUÇÃO AOS CÓDIGOS CÍCLICOS

Os códigos cíclicos binários são uma importante subclasse de códigos de bloco lineares. São códigos de fácil implementação com registradores de deslocamento realimentados. O cálculo da síndrome também pode ser facilmente executado de forma similar, com registradores de deslocamento realimentados.

Um código linear (n, k) é chamado de código cíclico se ele pode ser descrito pela propriedade apresentada a seguir. Se a *n-tupla*

$$\mathrm{c} = (c_0, c_1, c_2, \ldots, c_{\mathrm{n-1}})$$

é um vetor código no subespaço S, então,

$$\mathrm{c}^{(1)} = (c_{n-1}, c_0, c_1, \ldots, c_{n-2}),$$

obtido pelo deslocamento correspondente a uma posição de bit, é também um vetor código em S.

Em geral,

$$\mathrm{c}^{(i)} = (c_{n-\mathrm{i}}, c_{n-i+1}, c_1, \ldots, c_{n-1}, c_0, c_1, \ldots, c_{n-i-1})$$

obtido pelo deslocamento correspondente a i posições de bit, é também um vetor código em S.

Os componentes de um vetor código u podem ser tratados como os coeficientes de um polinômio c(X) como mostrado a seguir.

$$\mathrm{c}(X) = c_0 + c_1X + c_2X^2 + \ldots + c_{n-1}X^{n-1} \qquad (2.42)$$

Nesta representação a presença ou ausência de cada termo no polinômio indica a presença de um 1 ou 0, respectivamente, na correspondente locação da *n-tupla*. Desta forma, o polinômio pode ter grau n - 1 ou menos.

Exemplo 2.17

Seja o vetor código

$$c = 0\ 0\ 0\ 1\ 1\ 0\ 1$$

de um código de bloco linear (7, 4). Sua representação polinomial é

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

ou seja, um polinômio de grau $n - 1$. O valor de $c^{(3)}$, que também pertence ao mesmo código (7, 4) é

$$c^{(3)} = 1\ 0\ 1\ 0\ 0\ 0\ 1.$$

* * *

Podem-se gerar códigos cíclicos usando um polinômio gerador da mesma forma que são gerados os códigos de bloco usando uma matriz geradora.

O polinômio gerador g(X) para um código cíclico (n, k) tem a forma

$$g(X) = g_0 + g_1 X + g_2 X^2 + ... + g_{n-k} X^{n-k} \qquad (2.43)$$

onde g_0 e g_{n-k} devem ser iguais a 1 e o grau do polinômio gerador deve ser $n - k$.
Finalmente, um polinômio g(X) é um polinômio gerador de um código cíclico (n, k) se, e somente se, ele for um fator de $X^n + 1$.

EXEMPLO 2.18

Verifique se o polinômio $X^3 + X + 1$ gera um código cíclico C = (7, 4).

Solução:

$X^7 + 1$	$X^3 + X + 1$
$(X^7 + X^5 + X^4)$	$X^4 + X^2 + X + 1$
$0 + X^5 + X^4 + 1$	
$(X^5 + X^3 + X^2)$	
$0 + X^4 + X^3 + X^2 + 1$	
$(X^4 + X^2 + X)$	
$0 + X^3 + X + 1$	
$(X^3 + X + 1)$	
0	

Conclusão: O polinômio $X^3 + X + 1$ gera um código cíclico (7, 4) e ainda permite-nos concluir que o polinômio $X^4 + X^2 + X + 1$, que é o quociente da divisão realizada, gera um código cíclico (7, 3), pois

$$(X^7 + 1) = (X^4 + X^2 + X + 1)\ (X^3 + X + 1)$$

* * *

A matriz geradora para um código cíclico gerado pelo polinômio gerador g(X) pode ser obtida fazendo

$$\mathrm{G}=\begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ & & \vdots & & & & \vdots & & & \\ 0 & \cdots & 0 & 0 & 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} \end{bmatrix} \tag{2.44}$$

Onde $g_0, g_1, g_2, \ldots, g_{n-k}$ são os termos do polinômio $g(X) = g_0 + g_1X + g_2X^2 + \ldots + g_{n-k}X^{n-k}$.

É possível fazer a codificação de forma sistemática, através de uma matriz geradora G' obtida a partir da matriz G. Para isso, conforme visto anteriormente, G' deve ter a forma

$$\mathbf{G'}=\begin{bmatrix}\mathbf{P} & \vdots & \mathbf{I}_k\end{bmatrix}=\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1,(n-k)} & 1 & 0 & \cdots & 0 \\ p_{21} & p_{22} & \cdots & p_{2,(n-k)} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{k1} & p_{k2} & \cdots & p_{k,(n-k)} & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{2.45}$$

Isso pode ser feito através de operações lineares com as linhas de G até que G' tome a forma desejada.

EXEMPLO 2.19

Determine o vetor código de um código cíclico (7, 4), correspondente a mensagem m = 1011, utilizando a matriz geradora na forma sistemática, obtida a partir do polinômio gerador

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

Solução:

Do exemplo anterior, a matriz geradora na forma não sistemática é

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

Por inspeção verifica-se que a primeira e a segunda linha estão corretamente posicionadas para a obtenção de uma matriz na forma sistemática.

A terceira linha da matriz pode ser obtida através da soma das linhas 1 e 3.

A quarta linha da matriz pode ser obtida somando-se as linhas 1, 2 e 4.

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

O vetor código na forma sistemática é obtido pela operação U = m.G', conseqüentemente

$$c = \mathrm{m}.\mathrm{G}' = 1011 \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$c = 1001011$$

* * *

É fácil concluir que uma vez obtida a matriz G' na forma sistemática, a obtenção da matriz verificadora de paridade H, do código *C* gerado por G' é imediata, pois

$$\mathbf{H} = \left[\mathbf{I}_{n-k} \ \vdots \ \mathbf{P}^T \right]$$

2.5.2. CODIFICAÇÃO SISTEMÁTICA DE CÓDIGOS CÍCLICOS COM REGISTRADORES DE DESLOCAMENTO DE (*n-k*) ESTÁGIOS

Um vetor mensagem pode ser escrito na forma polinomial como

$$\mathrm{m}(X) = m_0 + m_1 X + m_2 X^2 + \ldots + m_{k-1} X^{k-1} \tag{2.46}$$

Na forma sistemática, os dígitos de mensagem são apresentados explicitamente como parte do vetor código. Para que a porção mensagem da palavra código ocupe as posições dos bits mais significativos, podemos fazer um deslocamento dos bits de mensagem para a direita, ficando as *n* - *k* posições mais a esquerda para a parte de paridade, ou seja,

$$X^{n-k}\ \mathrm{m}(X) = m_0 X^{n-k} + m_1 X^{n-k+1} + \ldots + m_{k-1} X^{n-1} \ . \tag{2.47}$$

Dividindo a expressão acima por g(*X*), obtém-se

$$X^{n-k}\ \mathrm{m}(X) = \mathrm{q}(X)\mathrm{g}(X) + \mathrm{r}(X) \tag{2.48}$$

ou, então

$$\mathrm{r}(X) + X^{n-k}\ \mathrm{m}(X) = \mathrm{q}(X)\mathrm{g}(X) = \mathbf{c}(X) \tag{2.49}$$

Onde o resto r(*X*) representa a parte de paridade do vetor código e o produto $X^{n-k}\ \mathrm{m}(X)$ representa a parte mensagem que foi deslocada *n* - *k* bits para a direita.

EXEMPLO 2.20

Determine o vetor código de um código cíclico (7, 4), na forma sistemática, para m = 1011, utilizando o polinômio gerador $g(X) = X^3 + X + 1$.

Solução:

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

$$X^{n-k}\, m(X) = X^3 (1 + X^2 + X^3) = X^3 + X^5 + X^6$$

Dividindo $X^{n-k}\, m(X)$ por $g(X)$ pode-se escrever

$$\underset{X^{n-k}(X)}{X^3 + X^5 + X^6} = \underset{q(X)}{(1 + X + X^2 + X^3)} \quad \underset{g(X)}{(1 + X + X^3)} + \underset{\text{resto}}{1}$$

Finalmente,

$$c(X) = r(X) + X^3\, m(X) = (1 + X^3 + X^5 + X^6)$$

$$c = 1001011$$

* * *

O circuito que faz as operações polinomiais apresentadas anteriormente está apresentado na Figura 2.11.

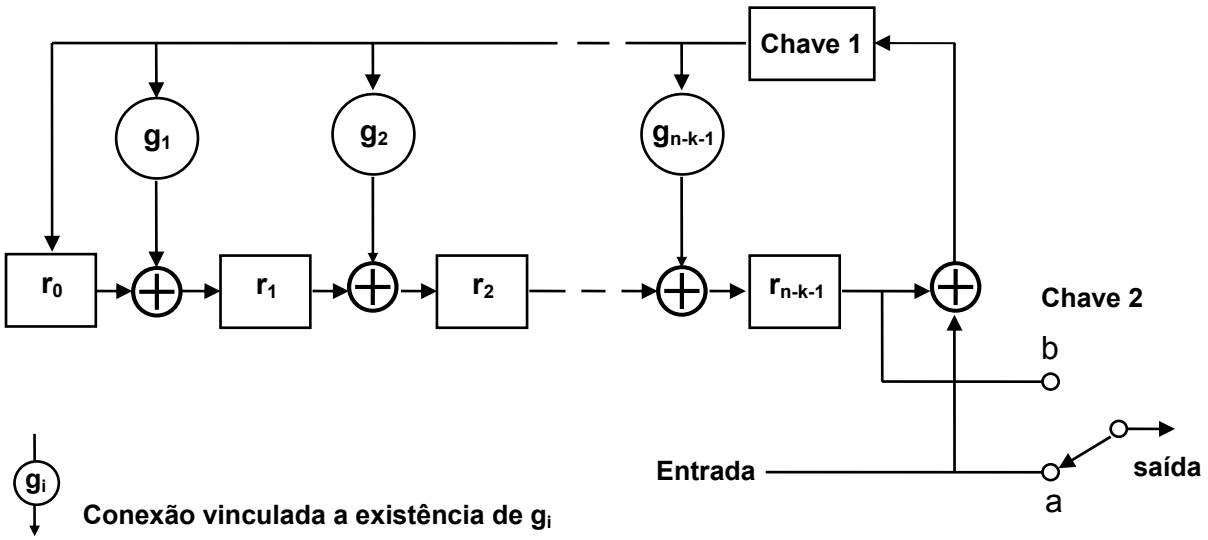

Figura 2.11 – Codificador para códigos cíclicos sistemáticos utilizando registradores de deslocamento.

O procedimento de codificação com o circuito da Figura 2.11 é o seguinte:

Passo 1: A chave 1 permanece fechada, para permitir a entrada dos bits de mensagem no estágio de codificação. A chave 2 permanece na posição (a) para permitir a transmissão dos bits de mensagem diretamente para o registro de saída, durante os primeiros *k* deslocamentos.

Passo 2: Após a transmissão dos *k* bits de mensagem a chave 1 é aberta (impedindo a realimentação) e a chave 2 é movida para a posição (b).

Passo 3: Os (*n*-*k*) bits de paridade que estão armazenados nos registros de deslocamento são transmitidos, completando a transmissão do polinômio código.

EXEMPLO 2.21

Seja o código (7, 4) cujo polinômio gerador é $g(X) = 1 + X + X^3$. Para o vetor mensagem m = 1011, o polinômio código resultante é $c(X) = 1 + X^3 + X^5 + X^6$, que corresponde ao vetor código c = 1001011. Mostre a formação e transmissão deste vetor código utilizando o circuito da Figura 2.11.

Solução:

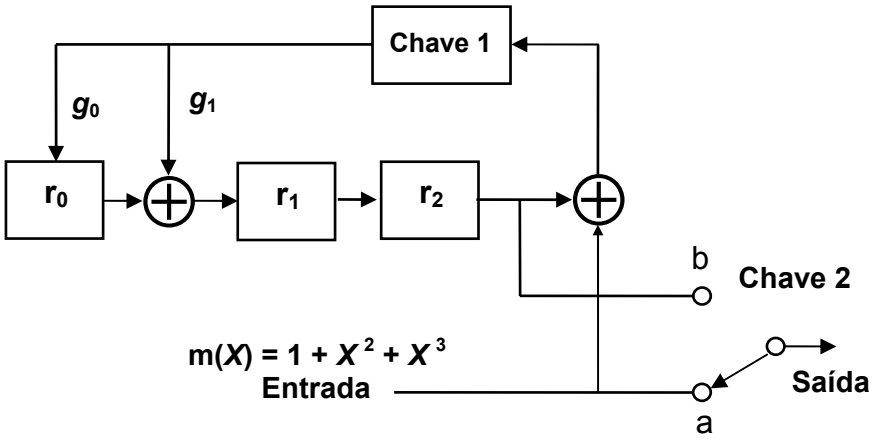

Figura 2.12 – Circuito de codificação para o Exemplo 2.21.

Fila de entrada	Número de deslocamentos	Conteúdo dos registradores	Saída
1011	0	000	-
101	1	110	1
10	2	101	1
1	3	100	0
-	4	100	1

Após os 4 deslocamentos a chave 1 é aberta, a chave 2 passa para a posição *b* e o conteúdo dos registros (paridade) é transmitido. Logo, o vetor transmitido é

c = 1001011.

* * *

2.5.3. DETECÇÃO DE ERROS DE CÓDIGOS CÍCLICOS SISTEMÁTICOS COM REGISTRADORES DE DESLOCAMENTO DE (*n-k*) ESTÁGIOS

Um vetor código transmitido c(X) pode ser alterado pela presença de ruído, de forma que função do decodificador é recuperar o vetor código transmitido a partir do vetor recebido.

Seja então um vetor recebido

$$\mathrm{r}(X) = r_0 + r_1 X + r_2 X^2 + \cdots + r_{n-1} X^{n-1} \tag{2.50}$$

O decodificador deve testar se o vetor recebido é um vetor código, o que equivale a dividir o polinômio recebido pelo polinômio gerador, pois

$$\mathrm{r}(X) = \mathrm{q}(X)\,\mathrm{g}(X) + \mathrm{S}(X) \tag{2.51}$$

Se a síndrome for zero, o vetor recebido é aceito como um vetor código, caso contrário, tem-se uma detecção de erro através da síndrome.

EXEMPLO 2.22

Determinar a síndrome do vetor r = 1001011, codificado na forma sistemática a partir do polinômio gerador g(X) = 1 + X + X^3 utilizando registradores de deslocamento. Mostre a formação da síndrome a cada deslocamento.

Solução:

O circuito para a determinação da síndrome é semelhante ao utilizado para a codificação, conforme mostrado a seguir.

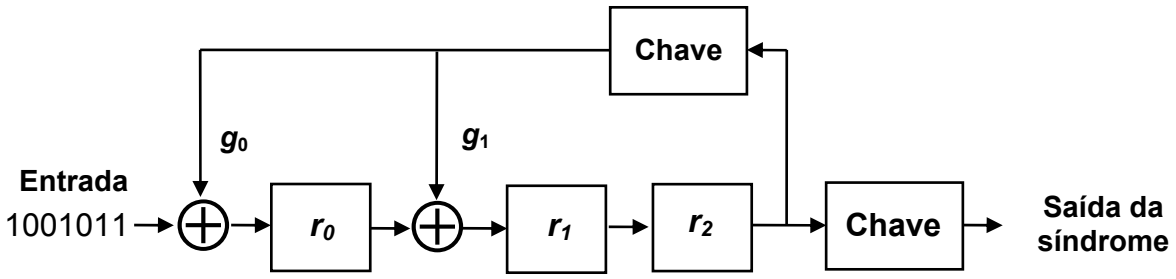

Figura 2.13 - Circuito de detecção de erros para o código cíclico gerado por g(X) = 1 + X + X^3.

Procedimento:

Passo 1: A chave 1 é inicialmente fechada e a chave 2 aberta. O vetor recebido é deslocado pela entrada dos registradores, cujos estados iniciais são todos zero. Após o vetor recebido estar todo nos registradores, seu conteúdo é a síndrome.

Passo 2: A chave 1 é então aberta e a chave 2 fechada, de forma a permitir que o vetor síndrome possa ser deslocado para fora dos registradores.

O conteúdo dos registradores a cada deslocamento é apresentado a seguir.

Fila de entrada	Número de deslocamentos	Conteúdo dos registradores
1001011	0	000
100101	1	100
10010	2	110
1001	3	011
100	4	011
10	5	111
1	6	101
-	7	000

* * *

2.5.4. DECODIFICADOR DE MEGGITT [3][4][6]

Códigos cíclicos podem ser decodificados utilizando-se o decodificador, cujo modelo genérico é apresentado na Figura 2.14, conhecido como *Decodificador de Meggitt*. Seu funcionamento pode ser descrito da seguinte forma:

Passo 1: Inicialmente as chaves CH 1, CH 3 e CH 4 estão fechadas e as chaves CH2 e CH5 abertas. A síndrome é gerada pelo deslocamento do vetor recebido, ao mesmo tempo em que este é armazenado no registrador de deslocamento.

Passo 2: A síndrome é lida pelo gerador de padrão de erro. O gerador de padrão de erro é um circuito combinacional que apresenta "1" em sua saída se e somente se a síndrome gerada corresponde ao padrão de erro na posição mais a direita do vetor recebido.

Se o vetor recebido for um vetor válido, o gerador de padrão de erro terá zero em sua saída e assim permanecerá até que o vetor recebido seja todo deslocado para fora.

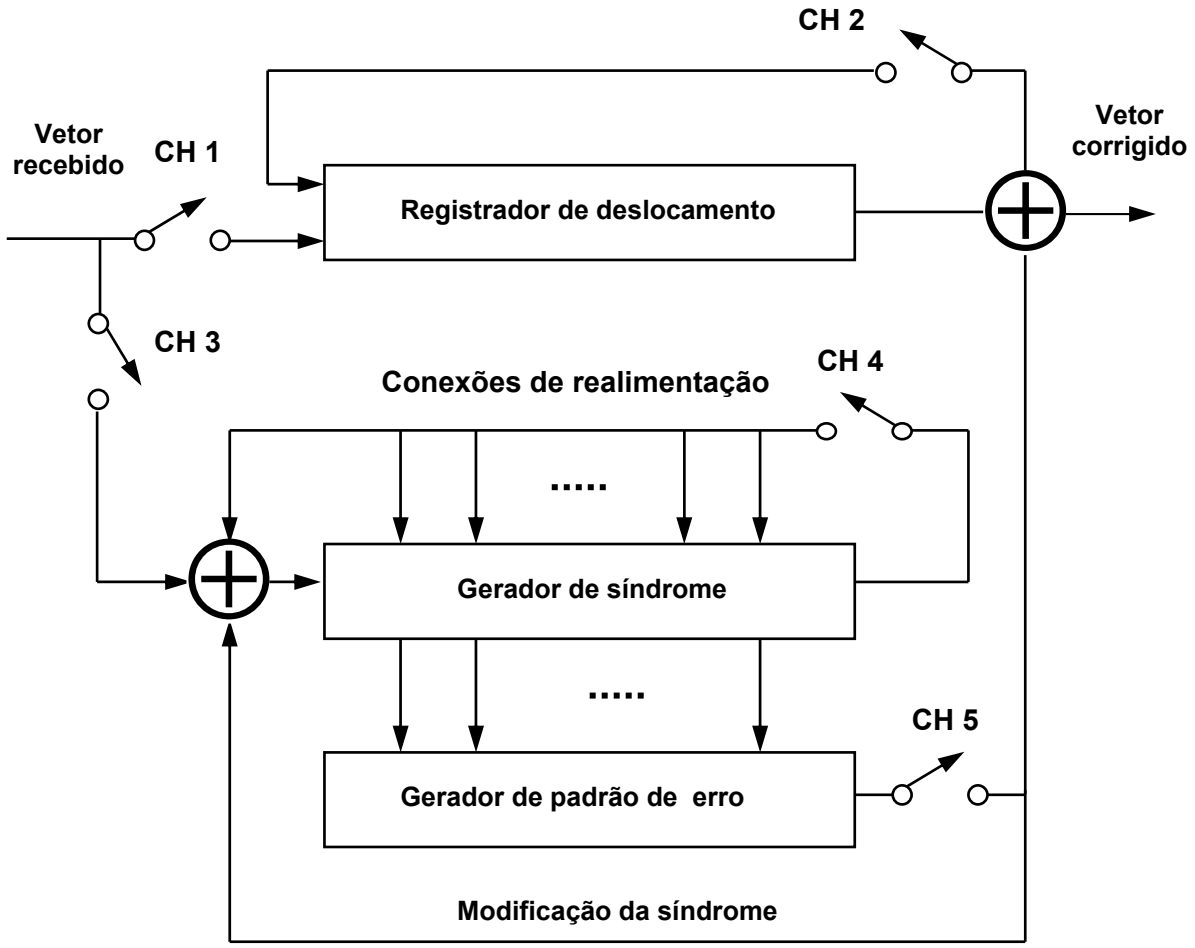

Figura 2.14 - Diagrama em blocos do decodificador de Meggitt.

Passo 3: As chaves CH 1 e CH 3 são abertas, a chave CH4 permanece fechada e as chaves CH2 e CH5 são fechadas. É iniciado o deslocamento cíclico do vetor recebido ao mesmo tempo em que o gerador de padrão de erro realimenta o gerador de síndrome.

Se o vetor recebido tiver um padrão de erro corrigível, durante o deslocamento cíclico do vetor recebido o padrão de erro irá se deslocar até que ele alcance a posição mais a direita do registrador de deslocamento. Durante esse processo, a síndrome é modificada a cada deslocamento. Quando o padrão de erro alcança a posição mais a direita do registrador de deslocamento, o gerador de padrão de erro apresentará "1" em sua saída que será somado na saída do decodificador e inverterá o bit correspondente a esta posição. Após o enésimo deslocamento, o conteúdo do registrador de deslocamento é o vetor recebido corrigido.

Passo 4: Após todo o deslocamento do vetor recebido, as chaves CH 1, CH 3 e CH 4 são fechadas e as chaves CH2 e CH5 abertas e a decodificação do próximo vetor é feita repetindo-se os passos de 1 a 3.

O exemplo apresentado a seguir ilustra todos os passos da operação de decodificação de um decodificador de Meggitt.

EXEMPLO 2.23

Considere a decodificação do código cíclico (7, 4) gerado pelo polinômio $g(X) = 1 + X + X^3$. Este é um código perfeito, capaz de corrigir exclusivamente todos os padrões de um erro. Suponha que o vetor código c = 1 0 0 1 0 1 1 tenha sido transmitido e o vetor recebido tenha sido r = 1 0 1 1 0 1 1. Evidentemente existe um erro na terceira posição da esquerda para a direita. Mostre a decodificação deste vetor, passo a passo, utilizando o decodificador de Meggitt.

Solução:

Os padrões de erros e suas respectivas síndromes para este código são apresentados na Tabela 2.9:

Tabela 2.9 – Padrões de erros e síndromes para o código gerado por $g(X) = 1 + X + X^2$

Padrões de erros	Síndromes
1000000	100
0100000	010
0010000	001
0001000	110
0000100	011
0000010	111
0000001	101

Note que a síndrome correspondente ao padrão de erro posicionado mais a direita (0000001) é 101 que será o único padrão de erro gerado pelo decodificador. O decodificador de Meggitt para este código está apresentado na Figura 2.15.

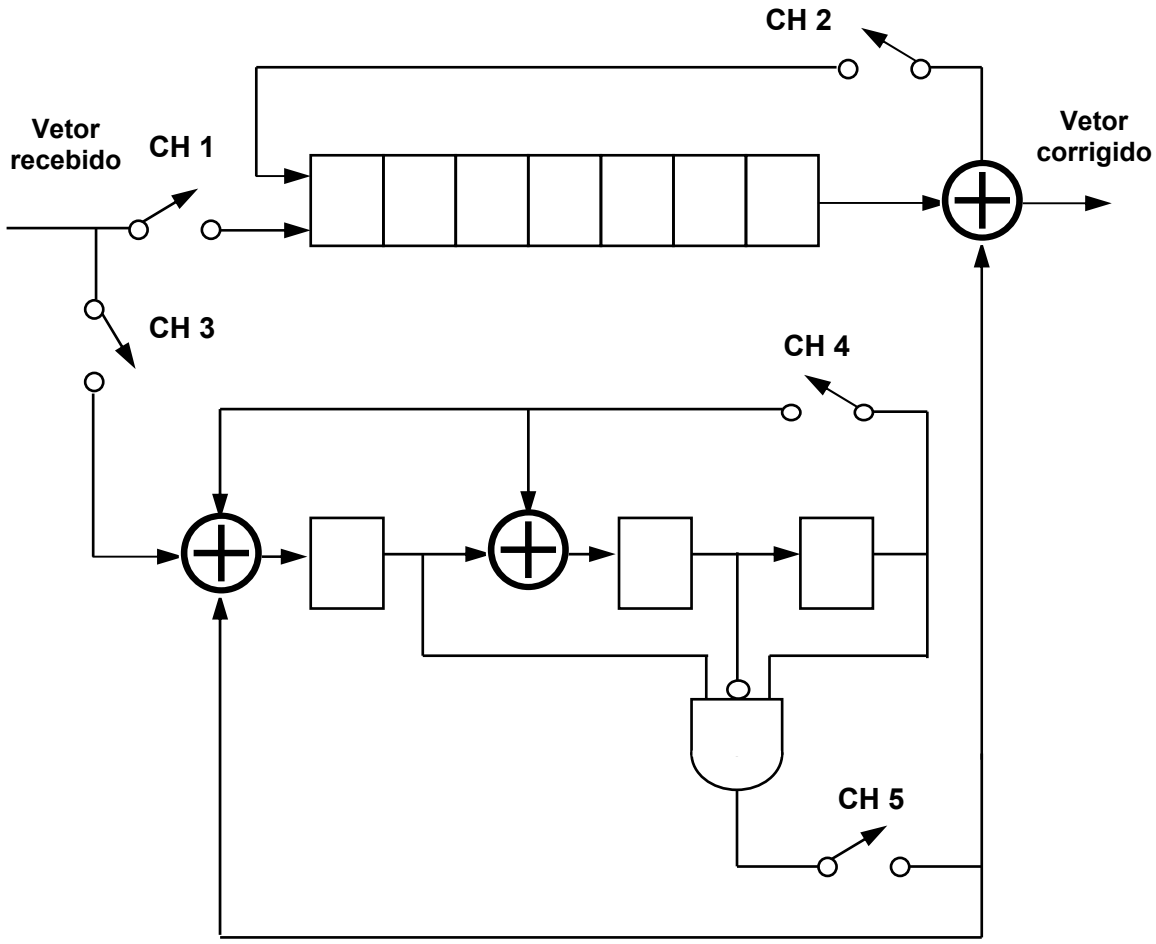

Figura 2.15 - Decodificador de Meggitt para o código cíclico gerado por $g(X) = 1 + X + X^2$.

Passo 1: Inicialmente as chaves CH 1, CH 3 e CH 4 estão fechadas e as chaves CH2 e CH5 abertas. O vetor 1 0 1 1 0 1 1 é carregado no registrador de deslocamento.

Passo 2: A síndrome correspondente ao padrão de erro é gerada e lida pelo gerador de padrão de erro, conforme mostrado na Figura 2.16 (a). Como a síndrome não corresponde ao padrão de erro na posição mais a direita do vetor recebido, a saída do gerador de padrão de erro é zero.

Passo 3: As chaves CH 1 e CH3 são abertas, CH 4 permanece fechada e as chaves CH2 e CH5 são fechadas. É iniciado o deslocamento cíclico do vetor recebido.

Note que enquanto a posição de erro não alcança a última posição do registrador de deslocamento, nada é somado à saída do decodificador. Veja Figura 2.16 (a) até (d).

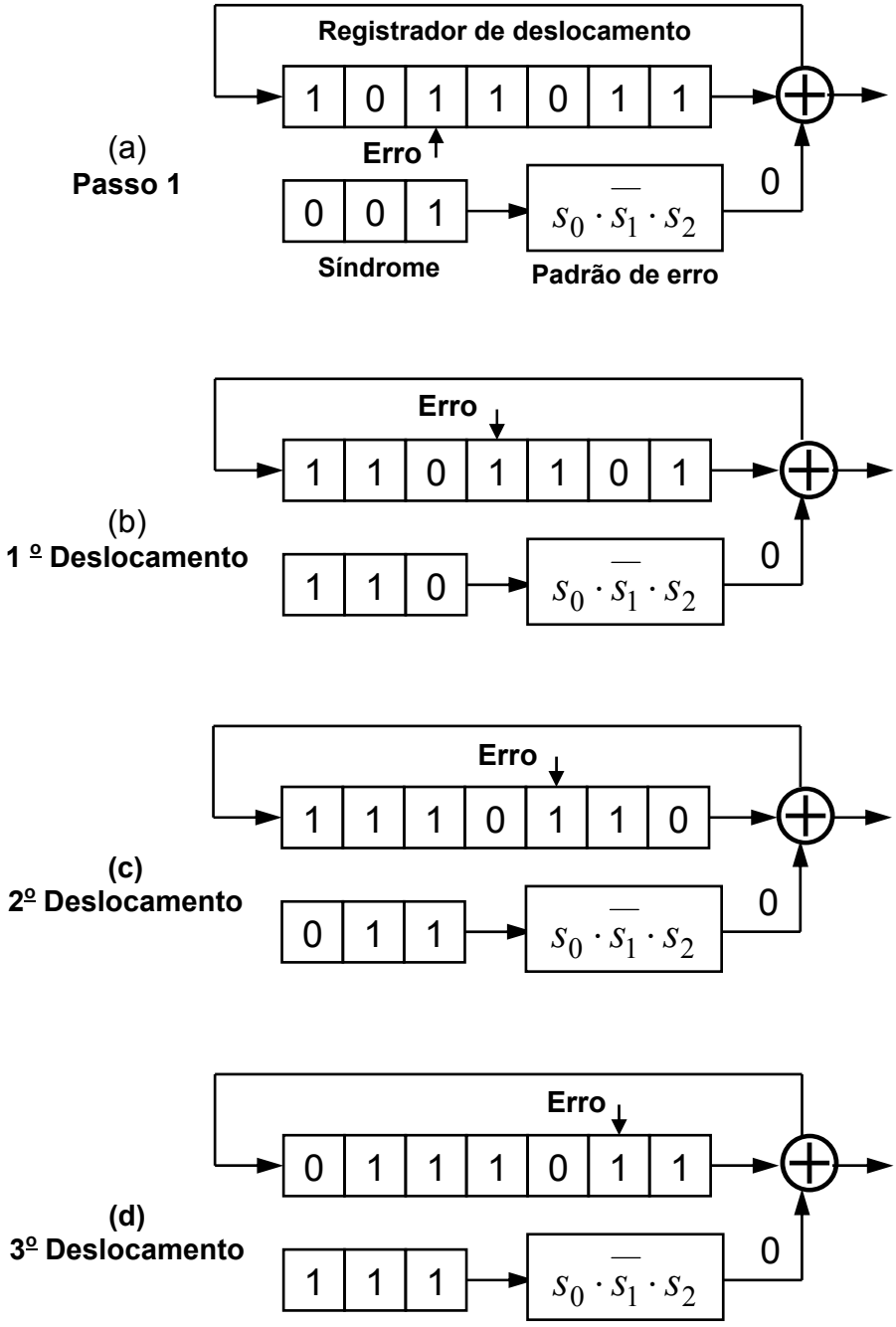

Figura 2.16 (a) a (d) - Decodificação passo-a-passo para o Exemplo 2.23

No quarto deslocamento, Figura 2.16 (e), o padrão de erro alcança a última posição do registrador de deslocamento e o gerador de padrão de erro gera "1" para ser somado com a última posição de bit do registrador de deslocamento, que é o bit errado.

A partir do quinto deslocamento, não há mais erro no vetor e a síndrome passa a ser zero e assim permanece até que todo o deslocamento cíclico seja completado. A Figura 2.16 (h) apresenta o vetor corrigido como conteúdo do registrador de deslocamento.

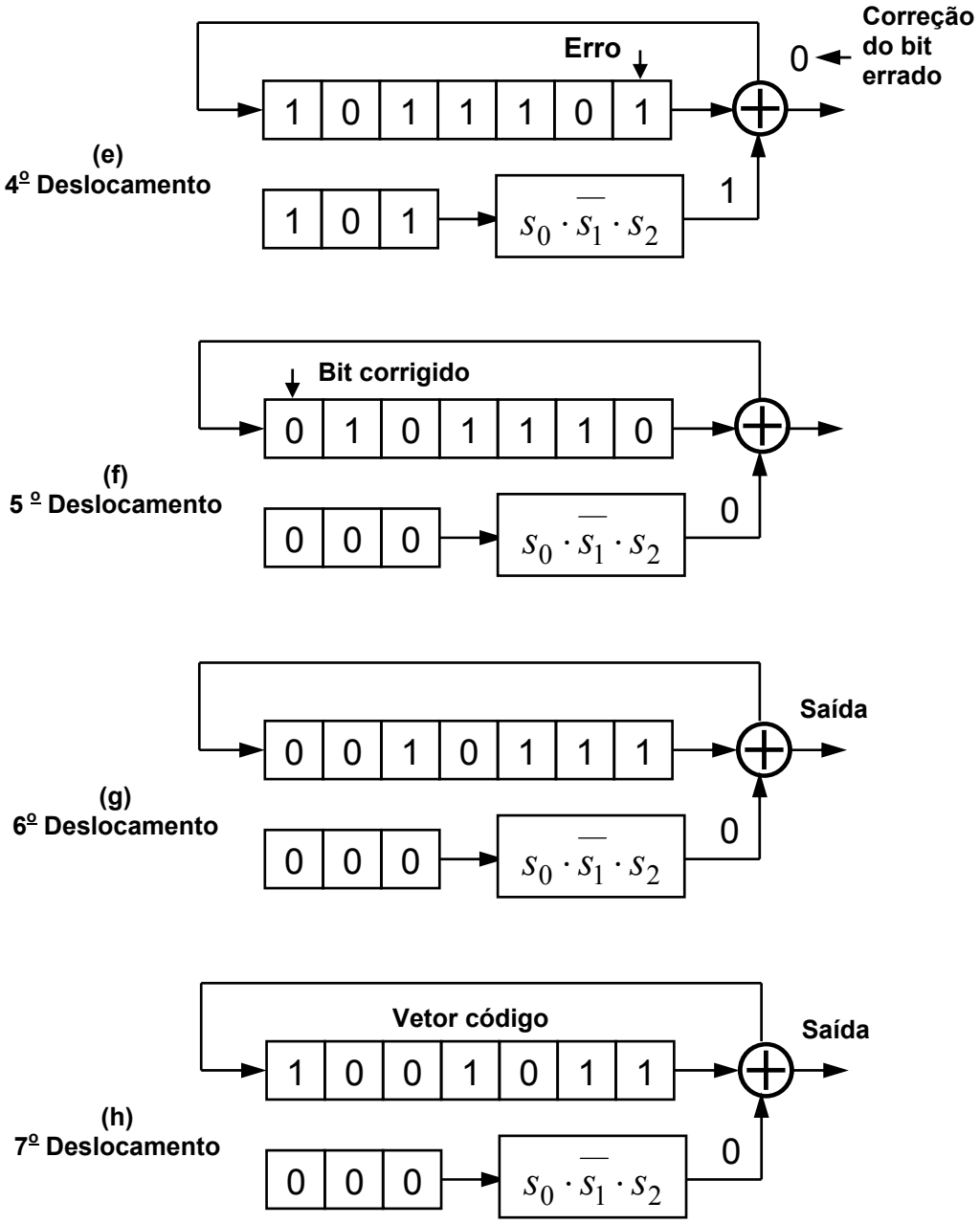

Figura 2.16 (e) a (h) - Decodificação passo-a-passo para o Exemplo 2.23

* * *

2.6. CÓDIGOS DE BLOCOS BEM CONHECIDOS [6]

Nas subseções a seguir são apresentados os códigos de blocos mais conhecidos, uma breve descrição de cada um e as suas principais características.

2.6.1. CÓDIGOS DE HAMMING

São códigos de bloco simples, que podem ser obtidos de forma cíclica, caracterizados pela seguinte estrutura:

$$(n, k) = (2^m - 1, 2^m - 1 - m) \tag{2.52}$$

onde $m = 2, 3, ...$ Isto é, suas características principais são apresentadas na Tabela 2.10.

Tabela 2.10 - Principais características dos Códigos de Hamming.

Comprimento do código:	$n = 2^m - 1$
Número de bits de informação:	$k = 2^m - 1 - m$
Número de bits de paridade:	$n - k = m$
Distância mínima:	$d_{min} = 3$
Capacidade de correção:	$t = 1$

Estes códigos têm uma distância mínima igual a 3 e apesar de terem capacidade de correção de erro limitada, eles pertencem a uma classe muito limitada de códigos de bloco conhecidos como códigos perfeitos.

2.6.2. CÓDIGOS GOLAY

Também é um código cíclico e perfeito. Possui maior capacidade de correção do que os códigos de Hamming. Suas principais características estão são apresentadas na Tabela 2.11.

$$(n, k) = (23, 12)$$

Tabela 2.11 - Principais características do Código Golay.

Comprimento do código:	$n = 23$
Número de bits de informação:	$k = 12$
Número de bits de paridade:	$n - k = 11$
Distância mínima:	$d_{min} = 7$
Capacidade de correção:	$t = 3$

O código Golay pode ser gerado a partir de um dos dois polinômios:

$$g_1(X) = 1 + X^2 + X^4 + X^5 + X^6 + X^{10} + X^{11} \tag{2.53}$$

$$g_2(X) = 1 + X + X^5 + X^6 + X^7 + X^9 + X^{11} \tag{2.54}$$

O código Golay é facilmente decodificado através de dois decodificadores de armadilha de erro refinados: o *decodificador de Kasami* e o *decodificador de busca sistemática*. Maiores detalhes a respeito de ambos podem ser encontrados em [4].

2.6.3. CÓDIGOS BCH (BOSE, CHAUDHURI & HOCQUENGHEM)

Os códigos BCH formam uma extensa classe de códigos cíclicos com grande capacidade de correção de erros. Eles são uma extraordinária generalização dos códigos de Hamming para correção de múltiplos erros. Os códigos BCH podem ser caracterizados da seguinte forma: para qualquer inteiro positivo m ($m \geq 3$) e t ($t < 2^{m-1}$), existe um código BCH binário com os parâmetros apresentados na Tabela 2.12.

Tabela 2.12 - Principais características dos códigos BCH.

Comprimento do código:	$n = 2^m - 1$
Número de bits de informação:	$k \geq 2^m - 1 - mt$
Número de bits de paridade:	$n - k \leq mt$
Distância mínima:	$d_{min} \geq 2t + 1$
Capacidade de correção:	t erros

Detalhes a respeito da implementação dos códigos BCH, bem como os principais algoritmos de decodificação são encontrados nas Referências [4] e [6].

2.6.4. CÓDIGOS REED-SOLOMON - *RS*

Os códigos Reed Solomon (RS) são uma sub-classe dos códigos BCH. São códigos cíclicos não binários com símbolos formados por seqüências de m bits, onde m é qualquer positivo inteiro tendo valor maior do que 2. Os códigos RS com símbolos de m bits existem para todo n e k para o qual

$$0 < k < n < 2^m + 2 \tag{2.55}$$

onde k é o número de símbolos de dados que estão sendo codificados e n é o número de símbolos códigos em um bloco codificado. As principais características dos códigos RS mais comuns estão apresentadas na Tabela 2.13.

Tabela 2.13 - Principais características dos códigos RS mais comuns.

Comprimento do código:	$n = 2^m - 1$
Número de bits de informação:	$k = 2^m - 1 - 2t$
Número de bits de paridade:	$n - k = 2t$
Distância mínima:	$d_{min} = n - k + 1$
Capacidade de correção:	$t = \left\lfloor \frac{n-k}{2} \right\rfloor$

Esses códigos, além de uma notável capacidade de correção de erros, são particularmente úteis para correção de erros em rajada. São extensamente utilizados em diversos sistemas de comunicações concatenados, principalmente, com códigos convolucionais além de outros sistemas de armazenamento de informações como CD para áudio digital.

2.6.5. CÓDIGOS REED-MULLER - *RM*

Os códigos *Reed-Muller* são uma importante subclasse dos códigos decodificáveis por lógica majoritária baseados em *geometria finita* ou *geometria euclidiana*. São códigos que podem ser gerados na forma cíclica e não cíclica. Suas principais características são apresentadas na Tabela 2.14.

Tabela 2.14 - Principais características dos códigos RM.

Comprimento do código:	$n = 2^m - 1$
Número de bits de informação:	$k = \sum_{i=0}^{\mu} \binom{m}{i}$
Número de bits de paridade:	$n - k = 2^m - 1 - \sum_{i=0}^{\mu} \binom{m}{i}$
Distância mínima:	$d_{min} = 2^{m-\mu} - 1$
Capacidade de correção:	$t = \left\lfloor \frac{2^{m-\mu} - 2}{2} \right\rfloor$

Para comprimentos moderados (n), os códigos RM possuem uma capacidade de correção de erro ligeiramente inferior que os códigos BCH. Entretanto, a decodificação por lógica majoritária é muito mais simples de implementar do que as decodificações para códigos BCH, o que torna atrativo o seu uso. Para comprimentos grandes, o desempenho dos códigos RM torna-se muito inferior quando comparado com os BCH.

2.7. EXERCÍCIOS

1. Projete um código de bloco linear (5, 2) com codificação na forma sistemática tendo como objetivo a maximização do valor de d_{min} e determine:

   a) A matriz geradora do código.
   b) A matriz verificadora de paridade.
   c) A capacidade de detecção e de correção de erros do código.
   d) A tabela de síndrome associada aos padrões de erros corrigíveis.

2. Considere um código sistemático (8, 4) cujas equações de verificação de paridade são:

$$p_0 = m_1 + m_2 + m_3,$$
$$p_1 = m_0 + m_1 + m_2,$$
$$p_2 = m_0 + m_1 + m_3,$$
$$p_3 = m_0 + m_2 + m_3,$$

   onde m_0, m_1, m_2 e m_3 são bits de mensagem e p_0, p_1, p_2 e p_3 são bits de verificação de paridade.

   Pede-se:

   a) Encontrar a matriz geradora e a matriz verificadora de paridade para este código.
   b) Mostre que a distância mínima deste código é 4. Justifique.
   c) Verifique se os vetores recebidos 10101010 e 01011100 são vetores códigos usando a síndrome de erros.
   d) Desenhe um circuito codificação para este código.
   e) Desenhe um circuito de decodificação para este código, de forma que a correção de todos os padrões de um erro e detecção simultânea de dois erros possa ser realizada.

3. Calcule a probabilidade de uma mensagem formada por uma seqüência de 12 bits codificada com um código de bloco linear (24, 12) possuir um erro. Admita que o código corrige todos os padrões de 1 e 2 erros e nenhum outro padrão a mais. Admita também que a probabilidade de erro do canal é igual a 10^{-3}.

4. Considere o código cíclico (7, 4) gerado por $g(X) = 1 + X^2 + X^3$

   a) Mostre que o polinômio gerador apresentado gera de faro um código cíclico (7, 4).
   b) Encontre todas as palavras códigos.
   c) Encontre a matriz verificadora de paridade do código.
   d) Verifique se o vetor recebido r = 1101101 é um vetor válido através da síndrome.
   e) Qual é a capacidade de correção de erros do código?
   f) Qual é a capacidade de detecção de erros do código?
   g) Desenhe um codificador para este código utilizando registradores de deslocamento.
   h) Desenhe o decodificador de Meggitt para este código.

5. Considere o código cíclico (15, 7) gerado por $g(X) = 1 + X^4 + X^6 + X^7 + X^8$.

a) Encontre a matriz verificadora de paridade deste código.
b) Determine a capacidade de correção de erros deste código.
c) Este é um código perfeito? Por quê?
d) Desenhe um circuito para o cálculo da síndrome de erros para este código.
e) Mostre como funciona o circuito para o cálculo da síndrome para cada bit de entrada do polinômio recebido $r(X) = 1 + X^{10} + X^{12} + X^{13} + X^{14}$.
f) Determine qual é o vetor código mais provável de ter sido o vetor transmitido, considerando o polinômio recebido apresentado na questão "e".

*** * ***

2.8. REFERÊNCIAS BIBLIOGRÁFICAS

[1] SKLAR, B., *Digital Communications: Fundamentals and Applications – 2nd ed.*, PTR Prentice Hall, Upper Saddle River, NJ, 2001. 1079 p.

[2] McELIECE, Robert J. *The theory of information and coding*. 2. ed. Cambridge: Cambridge University Press, 2002. ISBN 0521000955.

[3] ZIEMER, R. E.; PETERSON, R. L. *Introduction to Digital Communication*. 2 ed. Upper Saddle River: Prentice Hall, 2001. ISBN 0138964815.

[4] LIN, S.; COSTELO JR, D. J. *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs: Prentice Hall, 1983. ISBN 013283796X.

[5] BERLEKAMP, Elwin R. *Algebric Coding Theory*. New York: McGraw-Hill, 1968.

[6] WICKER, S. B. *Error control systems for digital communication and storage*. Upper Saddle River, New Jersey: Prentice Hall, 1995.