



III-2

Cyclic Redundancy Check

Comunicações

(29 Dezembro de 2008)



Sumário

1. Códigos cíclicos
2. Polinómio gerador
3. CRC
 1. Cálculo dos bits de paridade
 2. Verificação dos bits de paridade
4. Divisão de polinómios
 1. Utilização do MATLAB
5. Aplicações
6. Bibliografia



Códigos Cíclicos

- Os códigos cíclicos são uma sub-classe dos códigos lineares de bloco
 - **Linear:** o vector nulo pertence ao código; a soma modular de duas palavras do código é ainda uma palavra do código
 - **Bloco:** todas as palavras têm a mesma dimensão de n bits
- Nos códigos cíclicos tem-se que qualquer rotação cíclica de qualquer ordem sobre uma palavra de código é ainda uma palavra de código
- Exemplo: código de bit de paridade par (3,2)

m	c
00	000
01	011
10	101
11	110



Códigos Cíclicos

- Os códigos lineares de bloco são estabelecidos pela matriz geradora \mathbf{G}
- A partir de k palavras de código geram-se as 2^k palavras que constituem o código

$$\mathbf{c} = \mathbf{m} \times \mathbf{G}$$

- \mathbf{c} é vector de dimensões $1 \times n$;
 - \mathbf{m} é vector $1 \times k$
 - \mathbf{G} é matriz $k \times n$;
 - No código cíclico todas as palavras $\mathbf{c}(\mathbf{X})$ são geradas a partir do **polinómio gerador** $g(\mathbf{X})$ do código
- $$\mathbf{c}(\mathbf{X}) = \mathbf{m}(\mathbf{X})g(\mathbf{X})$$
- Todas as palavras de código são obtidas através do polinómio gerador



Códigos Cíclicos

- Tem-se $c(X) = m(X)g(X)$ em que:
 - $c(x)$ é a palavra de código – polinómio de grau $n-1$
 - $m(x)$ depende da mensagem – polinómio de grau $k-1$
 - $g(x)$ – polinómio gerador de grau q
- As palavras de código $c=[c_{n-1} \ c_{n-2} \ \dots \ c_1 \ c_0]$ podem ser analisadas como polinómios:
 - $c(X) = c_{n-1} X^{n-1} + c_{n-2} X^{n-2} + \dots + c_1 X + c_0$
- O número de bits redundantes (de paridade) corresponde ao grau do polinómio gerador



Polinómio Gerador

- Determinado polinómio $g(X)$ de grau q é gerador de um código (n,k) , com $q=n-k$, caso seja factor de X^{n+1}
- Ser factor de X^{n+1} implica que $\text{resto} \left[\frac{X^{n+1}}{g(X)} \right] = 0$
- Assim, a factorização do polinómio X^{n+1} é importante, neste contexto
- Através desta factorização, conseguimos obter polinómios geradores para códigos de diferentes dimensões



Factorização de $X^n + 1$

X^n+1	Factorização
X^3+1	$=(X+1)(X^2+X+1)$
X^5+1	$=(X+1)(X^4 + X^3 + X^2+X+1)$
X^7+1	$=(X+1)(X^3 + X^2+1) (X^3 + X+1)$
X^9+1	$=(X+1)(X^2 + X+1) (X^6 + X^3 +1)$
.....

Exemplos:

$X+1$, gera código (3,2)

$X^2 + X+1$, gera código (3,1)

$X+1$, gera código (5,4)

$(X^4 + X^3 + X^2+X+1)$, gera código (5,1)

$X^3 + X^2+1$ gera código (7,4)

$X^3 + X+1$ gera código (7,4)



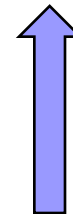
Polinómios Geradores

Código	Polinómio gerador $g(X)$
CRC4	$X^4+X^3+X^2+X+1$
CRC7	$X^7+X^6+X^4+1$
CRC12	$X^{12}+X^{11}+X^3+X^2+X+1$
CRC16	$X^{16}+X^{15}+X^2+1$
CRC-CCITT	$X^{16}+X^{12}+X^5+1$
CRC32	$X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$



Polinómios Geradores (outros exemplos)

Type	n	r	k	R_c	d_{\min}	t	$G(p)$						
Hamming codes <small>$r \geq 3$ $r = n - k$</small>	7	3	4	0.57	3	1				1	011		
	15	4	11	0.73	3	1				10	011		
	31	5	26	0.84	3	1				100	101		
BCH codes	15	8	7	0.46	5	2				111	010	001	
	31	10	21	0.68	5	2				11	101	101	001
	63	18	45	0.71	7	3	1	111	000	001	011	001	111
Golay code	23	11	12	0.52	7	3				101	011	100	011

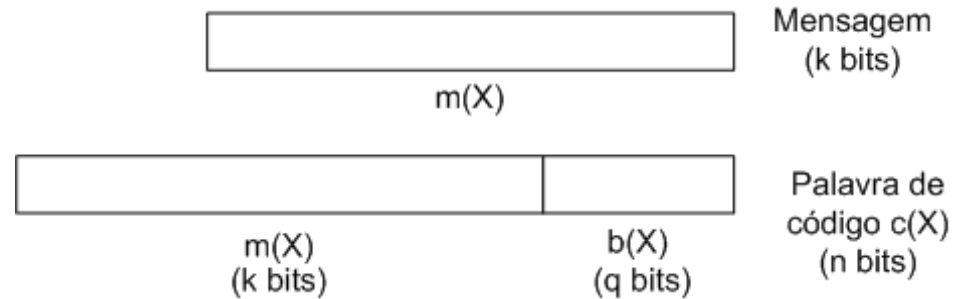


Polinómios Geradores



CRC – Cyclic Redundancy Check

- Num código cíclico sistemático, as palavras têm a seguinte organização



- Os bits $b(X)$, que constituem um polinómio de grau $q-1$ designam-se por **CRC-Cyclic Redundancy Check**
- A palavra de código é dada por

$$c(X) = m(X)X^q + b(X) = m(X)X^q + \text{resto} \left[\frac{m(X)X^q}{g(X)} \right]$$



CRC – Cyclic Redundancy Check

- O CRC resulta do resto da divisão de polinómios entre:
 - A mensagem deslocada de q bits para a esquerda
 - O polinómio gerador do código

$$CRC = b(X) = \text{resto} \left[\frac{m(X)X^q}{g(X)} \right]$$

- Dado que $g(X)$ tem grau q , resulta que $b(X)$ terá grau $q-1$, sendo constituído por q bits
- Assim, temos palavra de código com n bits (k de mensagem e q de paridade)



CRC – Cyclic Redundancy Check

- Exemplo de cálculo do CRC para código (7,4)
 - $m(X) = X^3 + 1 = [1\ 0\ 0\ 1]$
 - $g(X) = X^3 + X^2 + 1 = [1\ 1\ 0\ 1]$

$$\begin{aligned}
 CRC = b(X) &= \text{resto} \left[\frac{m(X)X^q}{g(X)} \right] = \text{resto} \left[\frac{(X^3 + 1)X^3}{X^3 + X^2 + 1} \right] = \text{resto} \left[\frac{X^6 + X^3}{X^3 + X^2 + 1} \right] \\
 &= X + 1
 \end{aligned}$$

$$\begin{array}{r}
 1001000 \\
 1101 \\
 01010 \\
 1101 \\
 01010 \\
 1101 \\
 1110 \\
 1101 \\
 \underline{011}
 \end{array}
 \quad
 \begin{array}{r}
 1101 \\
 \hline
 1111
 \end{array}$$

$$\begin{aligned}
 c(X) &= m(X)X^3 + b(X) = (X^3 + 1)X^3 + (X + 1). \\
 &= X^6 + X^3 + X + 1 \\
 &= [1001\ 011]
 \end{aligned}$$



CRC – Cyclic Redundancy Check

- Exemplo de cálculo do CRC para código (7,4)

- $m(X) = X^3 + 1 = [1 \ 0 \ 0 \ 1]$

- $g(X) = X^3 + X^2 + 1 = [1 \ 1 \ 0 \ 1]$

$$CRC = b(X) = \text{resto} \left[\frac{X^6 + X^3}{X^3 + X^2 + 1} \right] = X + 1$$

- Em MATLAB: uso da função *deconv* que realiza a divisão de polinómios

```
>> mq = [1 0 0 1 0 0 0];  
>> g = [1 1 0 1];  
>> [q, r] = deconv( mq, g);  
>> mod(q,2)  
ans =  
    1    1    1    1  
>> mod(r,2)  
ans =  
    0    0    0    0    0 1 1
```



CRC – Cyclic Redundancy Check

- O decodificador, em modo de detecção calcula o síndrome $s(X)$
- Dado que $c(X)=m(X)g(X)$, tem-se que qualquer palavra de código é factor do polinómio gerador
- Seja $y(X) = c(X) + e(X)$ a palavra recebida, em que $e(X)$ é o padrão de erro
 - Caso $e(X)$ seja nulo o síndrome é nulo

$$s(X) = \text{resto} \left[\frac{y(X)}{g(X)} \right] = \text{resto} \left[\frac{c(X)}{g(X)} \right] = \text{resto} \left[\frac{m(X)g(X)}{g(X)} \right] = 0$$

- Caso $e(X)$ seja não nulo o síndrome é não nulo e depende do valor de $e(X)$

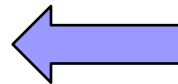
$$s(X) = \text{resto} \left[\frac{y(X)}{g(X)} \right] = \text{resto} \left[\frac{m(X)g(X) + e(X)}{g(X)} \right] = \text{resto} \left[\frac{e(X)}{g(X)} \right]$$



CRC – Cyclic Redundancy Check

- Na descodificador temos divisão de polinómios $s(X) = \text{resto} \left[\frac{c(X)}{g(X)} \right]$
- Recorrendo ao MATLAB, podemos usar a função *deconv*
- Sejam $c(X) = X^6 + X^3 + X + 1$ $g(X) = X^3 + X^2 + 1$
 $= [1001\ 011]$ $= [1101]$

```
>> c = [1 0 0 1 0 1 1];  
>> g = [1 1 0 1];  
>> [q, s] = deconv(c, g);  
>> mod(s,2)  
ans =  
    0    0    0    0    0    0    0
```



Síndroma nulo
Ausência de erros



CRC – Cyclic Redundancy Check

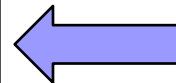
- Introduzindo 1 erro no penúltimo bit na palavra $c(X)$ temos

$$\begin{aligned}y(X) &= c(X) + e(X) = (X^6 + X^3 + X + 1) + (X) \\ &= X^6 + X^3 + 1 \\ &= [1001\ 001]\end{aligned}$$

$$\begin{aligned}g(X) &= X^3 + X^2 + 1 \\ &= [1101]\end{aligned}$$

```
>> c = [1 0 0 1 0 0 1];  
>> g = [1 1 0 1];  
>> [q, s] = deconv(c, g);  
>> mod(s,2)
```

```
ans =  
  0  0  0  0  0  1  0
```



Síndrome não nulo
Erros detectados



CRC - *Cyclic Redundancy Check*

- Tipicamente é utilizado em modo de **detecção** de erros
- Quando a distância mínima do código for maior ou igual a 3, também pode ser usado em modo **correção**
- Tipicamente temos um número reduzido de bits de paridade calculado para elevado número de bits de mensagem
 - $n \gg q > 1$
- O CRC tem elevada capacidade de detecção de erros, especialmente de *burst* de erros (rajada de erros)
- Um *burst* ou rajada de erros define-se como um bloco contíguo de bits recebidos em erro; o primeiro e último bit distam B bits entre si, sendo B o comprimento do *burst*



CRC - *Cyclic Redundancy Check*

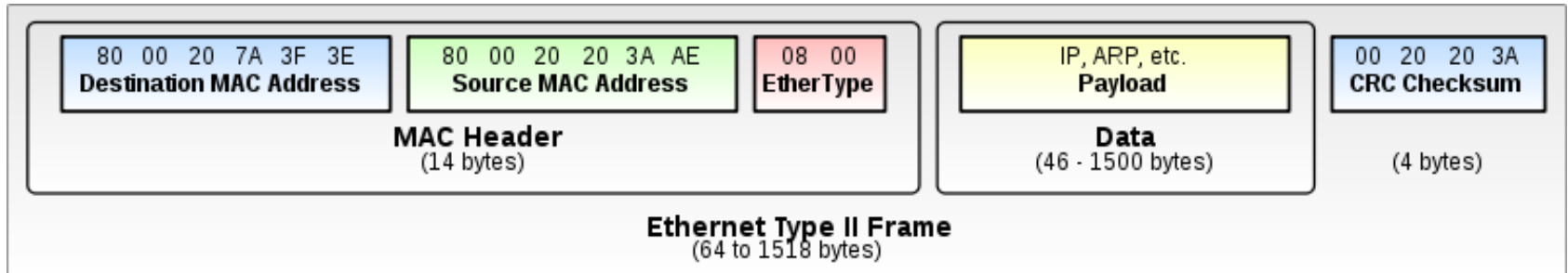
- Elevada capacidade de detecção de erros:
 - todos os *burst* de dimensão q ou menor
 - uma fracção dos *burst* de dimensão $q+1$; a fracção é $1-2^{-(q-1)}$
 - uma fracção dos *burst* de dimensão superior a $q+1$; a fracção é $1-2^{-q}$
 - todas as combinações de d_{\min} ou menos erros
 - todos os padrões com número ímpar de erros, quando o gerador tem número par de coeficientes não nulos

- Por exemplo, para o código CRC7 com $g(X)=X^7+X^6+X^4+1$ temos
 - todos os *burst* de dimensão 7 ou menor
 - $1-2^{-(q-1)} = 1 - 2^{-(7-1)} = 98,44\%$ dos *burst* de dimensão 8
 - $1-2^{-q} = 1 - 2^{-(7)} = 99,22\%$ dos *burst* de dimensão superior a 8
 - todos os padrões com número ímpar de erros



Aplicações

- Norma Ethernet 802.3 (Rede Local - LAN)
- Usa CRC32 (32 bits / 4 bytes) para verificação da integridade da trama; $g(X)=X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$
- O campo *FCS-Frame Check Sequence* no header da trama tem sempre 32 bits, independentemente da dimensão da trama
- A dimensão máxima da trama é 1518 bytes (12144 bits)



Aplicações

- Norma Ethernet 802.3

$$g(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

- Existem sempre 32 bits de paridade
- A trama tem dimensão mínima e dimensão máxima; esta última é 1518 bytes (12144 bits); temos um código $(n, n-32)$

Análise da distância mínima em função da dimensão da trama n

n	d_{\min}
3007 – 12144	4
301-3006	5
204-300	6
124-203	7
90-123	8

Fonte:

J. Moreira and P. Farrell, **Essentials of Error-Control Coding**, 2006, John Wiley and sons.

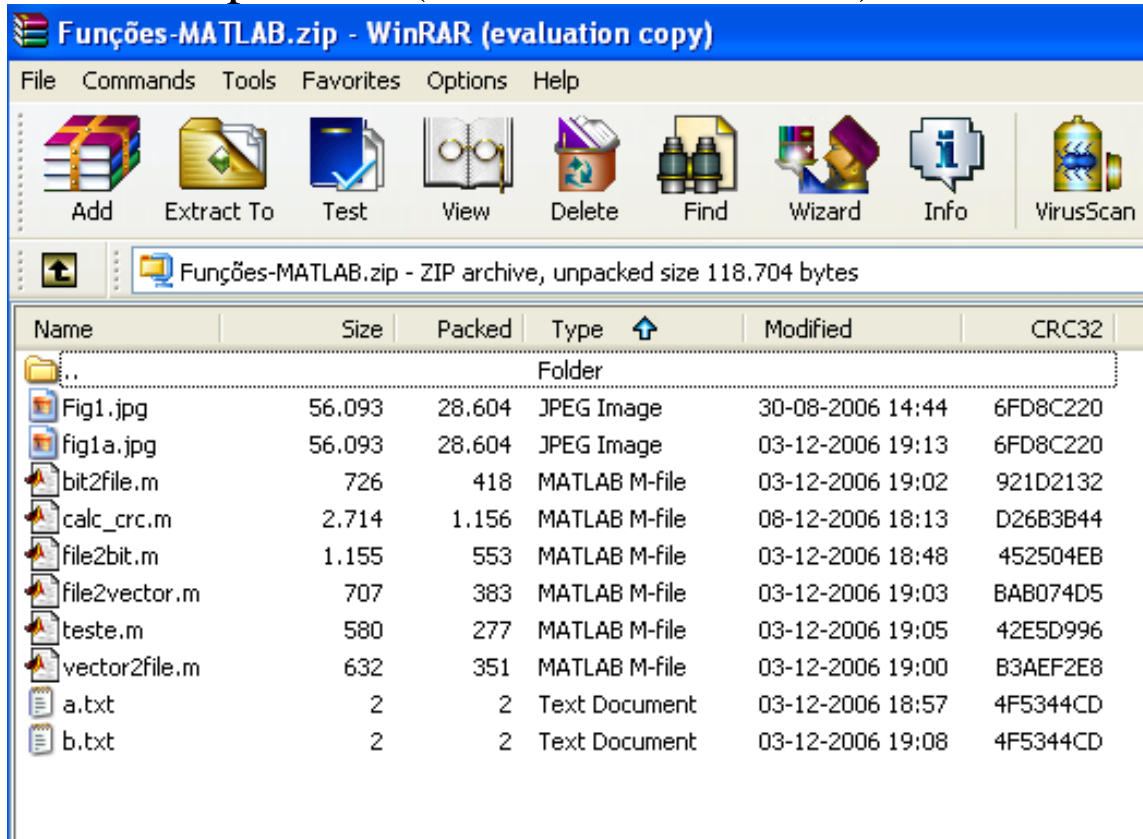
[Pág. 93]



Aplicações

■ Codificador de fonte WinRAR

- ❑ Usa CRC32 para verificação da integridade de cada ficheiro comprimido
- ❑ Antes de descomprimir o ficheiro, verifica a integridade do ficheiro comprimido (cálculo do síndrome)



The screenshot shows the WinRAR interface for a ZIP archive named 'Funções-MATLAB.zip'. The interface includes a menu bar (File, Commands, Tools, Favorites, Options, Help) and a toolbar with icons for Add, Extract To, Test, View, Delete, Find, Wizard, Info, and VirusScan. Below the toolbar, the archive's contents are listed in a table. The table has columns for Name, Size, Packed, Type, Modified, and CRC32. A blue arrow points from the text 'CRC32' on the right to the CRC32 column header in the table.

Name	Size	Packed	Type	Modified	CRC32
..			Folder		
Fig1.jpg	56.093	28.604	JPEG Image	30-08-2006 14:44	6FD8C220
fig1a.jpg	56.093	28.604	JPEG Image	03-12-2006 19:13	6FD8C220
bit2file.m	726	418	MATLAB M-file	03-12-2006 19:02	921D2132
calc_crc.m	2.714	1.156	MATLAB M-file	08-12-2006 18:13	D26B3B44
file2bit.m	1.155	553	MATLAB M-file	03-12-2006 18:48	452504EB
file2vector.m	707	383	MATLAB M-file	03-12-2006 19:03	BAB074D5
teste.m	580	277	MATLAB M-file	03-12-2006 19:05	42E5D996
vector2file.m	632	351	MATLAB M-file	03-12-2006 19:00	B3AEF2E8
a.txt	2	2	Text Document	03-12-2006 18:57	4F5344CD
b.txt	2	2	Text Document	03-12-2006 19:08	4F5344CD

CRC32



Bibliografia

□ Folhas de apoio

- A. Ferreira, [Códigos detectores e correctores de erros](http://www.deetc.isel.ipl.pt/sistemastele/cm/), disponível na página da unidade curricular <http://www.deetc.isel.ipl.pt/sistemastele/cm/>

□ Livro

- J. Moreira and P. Farrell, **Essentials of Error-Control Coding**, 2006, John Wiley and sons.

