Nível Ligação de Dados (*Data Link Layer*)

- <u>Como se detectam tramas com erros?</u>

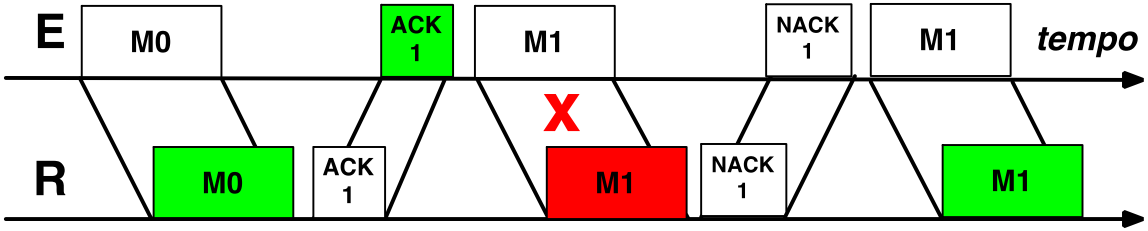
<u>Códigos</u>
e
<u>Redundância</u>

Mensagem (*m*=*1*)	Palavra código (*n*=*2*)
0	00
1	11

- <u>Como se corrigem tramas detectadas como tendo erros?</u>

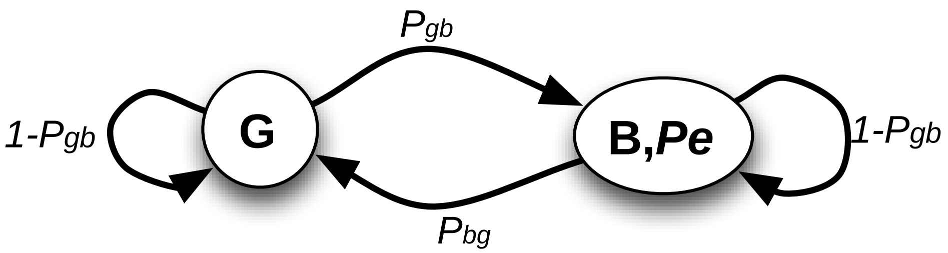
Pelo receptor **<u>(FEC)</u>**

Pelo emissor **<u>(ARQ)</u>**



Detecção de Erros

- **Tipos de erros:**
  - "**Simples**" e "Em rajada (*burst*)"



- **Técnicas para detecção:**
  - Eco no receptor
    - e.g., Terminal com eco
  - Envio por dois caminhos distintos (diversidade)
  - **Informação redundante**:
    - Bit de paridade (paridade horizontal, vertical e cruzada)
    - Código de bloco
    - Código polinomial (CRC)

Correcção de Erros

- ***Forward Error Correction*** **- FEC**
  - Correção dos erros por parte do receptor através de informação **redundante** recebida.
  - Exemplo:
    - Códigos de Blocos (e.g., Código de Hamming – corrige 1 erro simples)

- ***Automatic Repeat Request*** **- ARQ**
  - Correção dos erros por parte do emissor através de repetição, envolve as seguintes fases:
    - 1º: Detecção dos erros pelo receptor usando informação **redundante** (e.g., CRC)
    - 2º: Indicação do receptor para o emissor das tramas certas/erradas
    - 3º: Retransmissão automática das tramas erradas pelo emissor
  - Exemplo:
    - Stop-and-Wait
    - Go-Back-N
    - Selective Repeat

Nível Ligação de Dados (*Data Link Layer*)

- Como se detectam/corrigem tramas com erros?
  - Tipos de erros de bit (recordar).
  - Códigos detectores/correctores de erros.
- Como comparar códigos para detectar/corrigir erros?
  - Taxa de redundância.
  - Distância de Hamming de um código.

Princípios fundamentais: Redundância

Para enviar *m* bits de mensagem, transmite-se $n = m + r$ bits, ***r* bits são redundantes**.

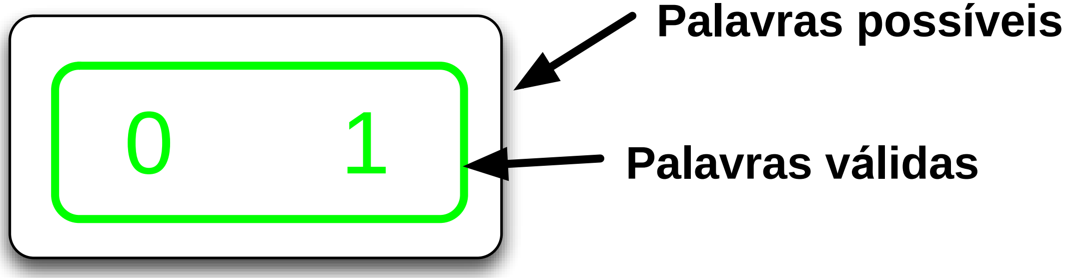
Taxa de redundância = r/n.

Taxa do código = m/n.

- Os ***n*** bits transmitidos definem as **2^n** **palavras possíveis** de serem recebidas.
- Os ***m*** bits de mensagem definem as **2^m** mensagens passíveis de serem transmitidas.
- A cada mensagem passível de ser transmitida corresponde uma palavra do código => Existem 2^m **palavras válidas** nesse código.

- E quando não há redundância? ($r=0$).

Exemplo $n=m=1$.



Conclusão:

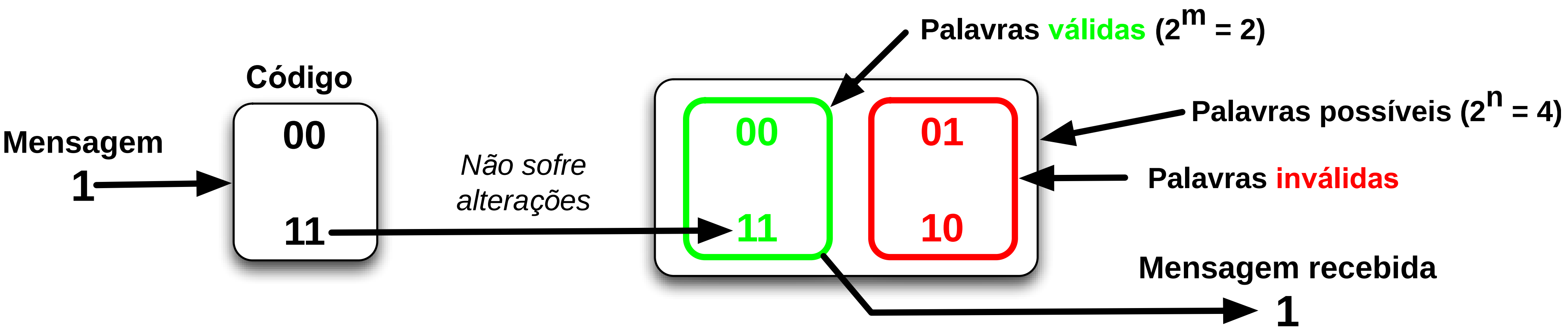
Todas as palavras possíveis são válidas, logo não se consegue detectar qualquer situação de erro.

Princípios fundamentais: Códigos

Mensagem (*m=1*)	Palavra código (*n=2*)
0	00
1	11

A cada mensagem passível de ser transmitida corresponde uma palavra do código.
=> Existem 2^m **palavras válidas** no código.

- Palavra válida do código não é transformada/alterada.



Palavras válidas (2^m = 2)

Código

00

Mensagem

1

Não sofre alterações

11

00

01

Palavras possíveis (2^n = 4)

Palavras inválidas

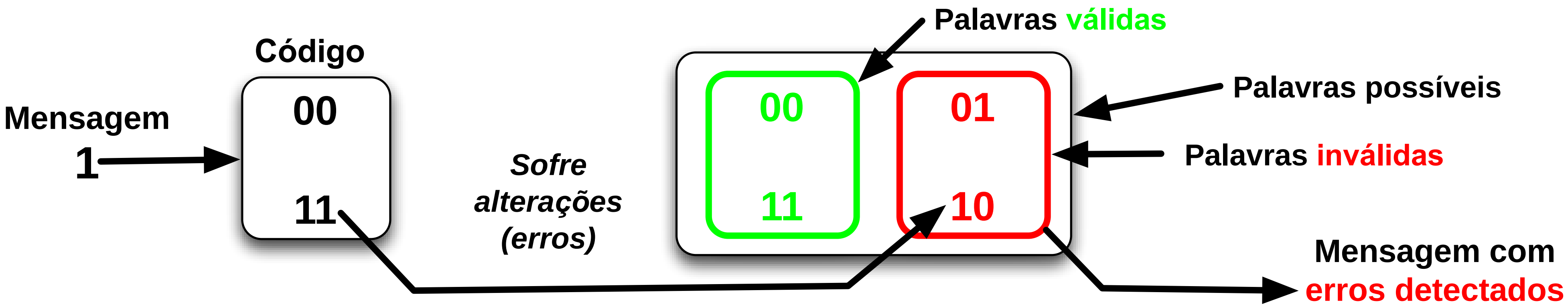
11

10

Mensagem recebida

1

- Palavra válida do código é transformada numa palavra não válida.



Palavras válidas

Código

00

Mensagem

1

11

Sofre alterações (erros)

00

01

Palavras possíveis

Palavras inválidas

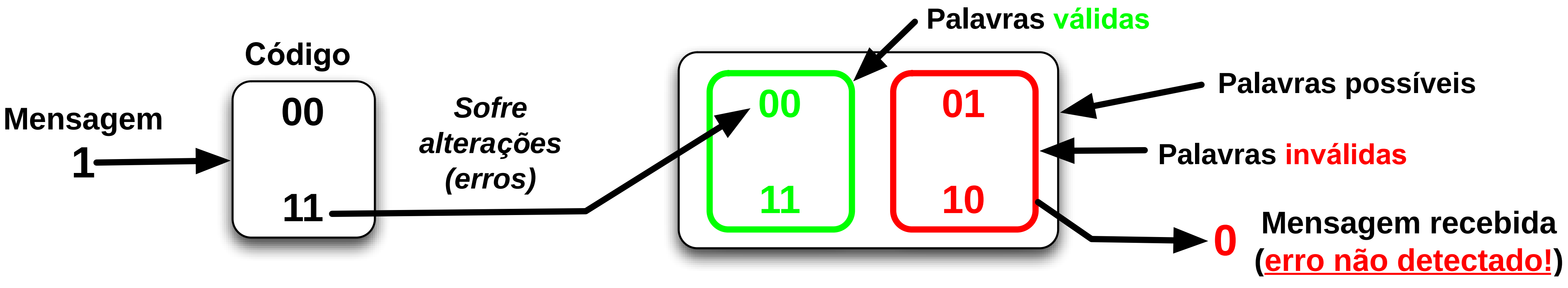
11

10

Mensagem com erros detectados

Princípios fundamentais: Código

- Palavra válida do código é transformada numa palavra válida.



Palavras válidas

Código

Mensagem

1

00

11

Sofre alterações (erros)

00

11

01

10

Palavras possíveis

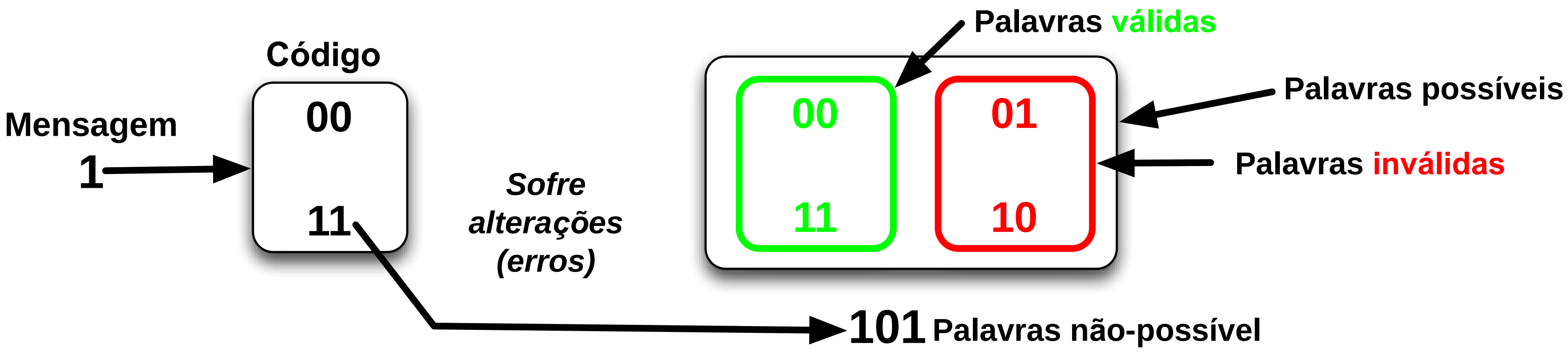
Palavras inválidas

0

Mensagem recebida

(erro não detectado!)

- Palavra válida do código é transformada numa palavra não possível. (Situação não considerada!)



Palavras válidas

Código

Mensagem

1

00

11

Sofre alterações (erros)

00

11

01

10

Palavras possíveis

Palavras inválidas

101

Palavras não-possível

Distância de Hamming

- ***Distância de Hamming entre duas palavras binárias*:**
  - **Definição:** Número de bits que se tem de alterar para passar de uma palavra para outra:

$$\begin{array}{r} 0110\ 1100 \\ \oplus\ 1100\ 1101 \\ \hline 1010\ 0001 \end{array} => d = 3$$

  - No caso particular em que se compara a sequência de bits transmitida, *T*, e recebida, *R*, a distância de Hamming representa o número de bits errados.

Richard Hamming (1915-1998) é um dos pioneiros no estudo das questões relacionadas com a detecção e correcção de erros.

A distância de Hamming representa uma ferramenta teórica importante na definição destes processos.

Distância de Hamming de um código

Mensagem	Palavra código
00	0000000000
01	0000011111
10	1111100000
11	1111111111

- 0000011111 => Palavra possível e válida (i.e. do código)
  (Estará correcta?)
- 0010011101 => Palavra possível mas não válida
  (Qual será a correcta?)

Distância de *Hamming* de um código, *d*:

- **Definição**: *Distância de* Hamming *mínima entre qq duas palavras do código*.

  No caso exemplo d = *5*.

- **Detecção de erro:** Não há nenhuma palavra válida à distância de Hamming *0* da palavra recebida, i.e., corresponde a uma palavra não válida
  - Consegue detectar **de certeza** até (*d*-1) bits errados, em situações com *d* ou mais erros não se pode ter a certeza quanto à sua detecção embora esta possa ocorrer.

    Ex: Recebe 1000011110 -> tem de certeza erros!

    Ex: Recebe 0000011111 -> deve estar certa!

- **Correcção:** Escolhe a palavra válida à menor distância de *Hamming* da palavra recebida
  - Consegue corrigir **correctamente** até (*d*-1)/2 bits errados.

    Ex: Envia 000001111 e recebe 1100111111 -> tem erros de certeza, a palavra mais próxima da recebida é 1111111111 => "correcção" errada!

Como comparar códigos para detecção e correção de erros ?

Mensagem	Palavra código
00	000
01	011
10	101
11	110

Taxa código 2/3.

Ou?

Mensagem	Palavra código
00	0000000000
01	0000011111
10	1111100000
11	1111111111

Taxa código 2/10.

Mensagem	Palavra código
0	00
1	11

Taxa código 1/2.

Ou?

Mensagem	Palavra código
0	00
1	10

Taxa código 1/2.

Nível Ligação de Dados (*Data Link Layer*)

- Exemplo de código para detectar tramas com erros?
  - Código "Bit de paridade".
  - Códigos de redundância cíclica (CRC).
- Exemplo de código para detectar e corrigir tramas com erros?
  - Códigos de Hamming.

Bit de Paridade (r=1)

Paridade Par

e.g., 10110101
101101011
100101011 (5 1s => erro!)

e.g., 00001111
000011110
000011101 ~~correcto~~

Paridade Ímpar

e.g., 10110101
101101010

e.g., 00000000
000000001

e.g., 11111111
111111111

- **Expressão para o cálculo do bit paridade par:**

$$P = X_1 \oplus X_2 \oplus X_3 \oplus \cdots \oplus X_m$$

- **Distância de *Hamming* do Código 'Bit de Paridade': *d* = 2**

Códigos de Hamming, *Ham(n,m)*

- Código com 2^m mensagens (*m* bit úteis) e *r* bits de verificação (redundantes): $n = m + r$

- Objectivo: ser capaz de corrigir correctamente 1 erro.
  Logo, para qualquer código de Hamming, $d = 3$

  Detecta de certeza até 2 erros

  Corrige correctamente 1 erro

- Ideia chave:
  Cada mensagem tem *n*+1 padrões associados *(alteração de 1 bit + palavra válida)*

- Quantos bits de redundância, *r*, são necessários?
  Condição entre *m*, *n* e *r*:

$$\left[2^m(n+1)\right] \leq 2^n \Leftrightarrow (m+r+1) \leq 2^r$$

- Cálculo de *r*:
  - *Aproximação:* $r \approx \log_2(m) + 1$
  - *Requer Verificação:* $r \geq \log_2(m + r + 1)$

Código de Hamming

- Estrutura para cálculo dos bits *r* (*de redundância ou paridade*)
  - Bit na posição $j = 2^i$ é bit de paridade P_j
  - Bit na posição $j \neq 2^i$ é bit de mensagem X_j

  Exemplo: $P_1 P_2 X_3 P_4 X_5 X_6 X_7$
- Cálculo dos bits de paridade, P_j (***emissor***)

$$P_1 = X_3 \oplus X_5 \oplus X_7$$
$$P_2 = X_3 \oplus X_6 \oplus X_7$$
$$P_4 = X_5 \oplus X_6 \oplus X_7$$

	P_4	P_2	P_1
X_3	0	1	1
X_5	1	0	1
X_6	1	1	0
X_7	1	1	1

- Cálculo dos ***check*** bits, C_j (***receptor***)

$$C_1 = (X'_3 \oplus X'_5 \oplus X'_7) \oplus P'_1$$
$$C_2 = (X'_3 \oplus X'_6 \oplus X'_7) \oplus P'_2$$
$$C_4 = (X'_5 \oplus X'_6 \oplus X'_7) \oplus P'_4$$

\- $C_1 = C_2 = C_4 = 0$ i) não há erros, ou
ii) há erro(s) não detectado(s)

\- $C_1 \neq 0$ ou $C_2 \neq 0$ ou $C_4 \neq 0$ ⇒ mensagem com erros
Bit a corrigir indicado pela soma dos índices dos *C* não nulos

Palavras válidas do código Ham(7,4), par (*m*=4, *r*=3), paridade par.

1	2	3	4	5	6	7	bit
0	0	0	0	0	0	0	
1	1	1	0	0	0	0	
1	0	0	1	1	0	0	
0	1	1	1	1	0	0	
0	1	0	1	0	1	0	
1	0	1	1	0	1	0	
1	1	0	0	1	1	0	
1	1	1	0	1	1	0	
1	1	0	1	0	0	1	
0	0	1	1	0	0	1	
0	1	0	0	1	0	1	
0	0	1	1	1	0	1	
1	0	0	0	0	1	1	
0	1	1	0	0	1	1	
0	1	0	1	1	1	1	
1	1	1	1	1	1	1	

- E.g., recebe 0100111
- Falhou a paridade (*check* bits) 2 e 4. logo o bit errado é o 6 (=2+4)
- A palavra CORRIGIDA é 0100101, e a mensagem CORRIGIDA é 0101

Série de Problemas nº 4 – Prob. II

Considere o método de correcção de erros baseado nos códigos de **Hamming (paridade par)**.

a) Indique a estrutura de uma trama para a transmissão de **4 bits úteis** e o tipo de código de Hamming utilizado.

b) Determine qual a taxa do código e a sua redundância.

c) Determine qual a palavra de código enviada para a transmissão da mensagem **1010**.

d) Verifique que é possível, utilizando um exemplo, corrigir a mensagem se esta sofrer um erro num dos seus bits.

e) f) g) e h) => *Trabalho autónomo.*

Códigos Polinomiais (*Cyclic Redundancy Code* – CRC)

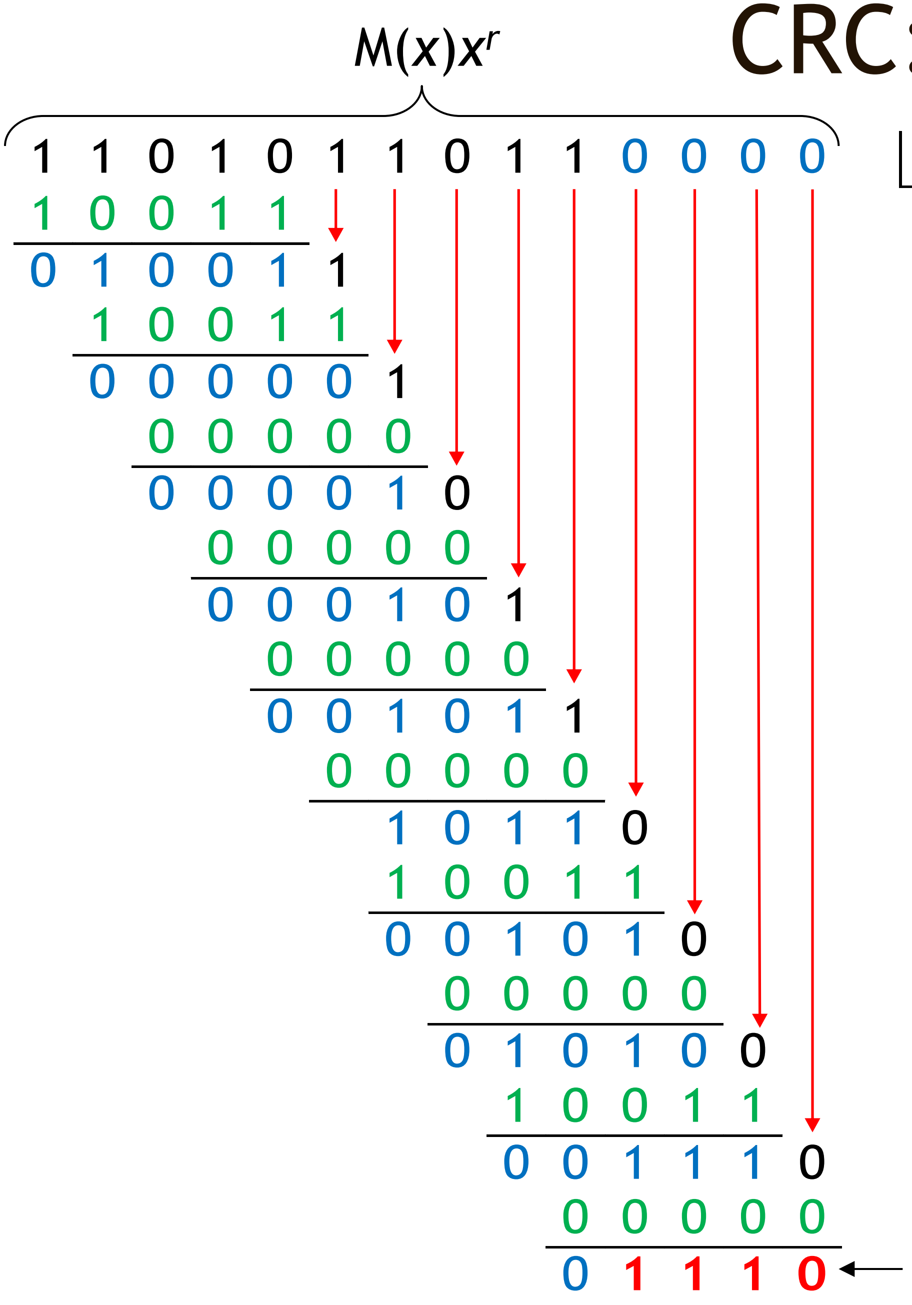
- Baseia-se na associação entre sequências binárias e polinómios.
- Os coeficientes do polinómio são os dígitos da sequência binária.
- As operações realizadas são em aritmética módulo 2.
- Exemplo (polinómio de M):

$$M = 101001 \ \text{->} \ M(x) = 1.x^5+0.x^4+1.x^3+0.x^2+0.x^1+1.x^0 = x^5+x^3+1$$

- Método:
  - Dada uma trama M com *m* bits;
  - O **emissor** gera uma sequência de *r* bits (Frame Check Sequence – FCS);
  - A trama resultante T com *m*+*r* bits tem de ser exactamente divisível por uma determinada sequência de bits (Polinómio gerador G);
  - O **receptor** divide a trama pela mesma sequência de bits e se o resto for zero assume que não há erros.

$$\boxed{FCS(x) = \text{Resto}\ [x^r M(x)/G(x)]}$$

CRC: Exemplo



$M(x)x^r$

```
1 1 0 1 0 1 1 0 1 1 0 0 0 0
1 0 0 1 1
0 1 0 0 1 1
  1 0 0 1 1
  0 0 0 0 0 1
    0 0 0 0 0
    0 0 0 0 1 0
      0 0 0 0 0
      0 0 0 1 0 1
        0 0 0 0 0
        0 0 1 0 1 1
          0 0 0 0 0
            1 0 1 1 0
            1 0 0 1 1
            0 0 1 0 1 0
              0 0 0 0 0
              0 1 0 1 0 0
                1 0 0 1 1
                0 0 1 1 1 0
                  0 0 0 0 0
                  0 1 1 1 0 ← Resto = FCS (4 bits)
```

```
| 1 0 0 1 1 ← G(x)
  1 1 0 0 0 0 1 0 1 0
```

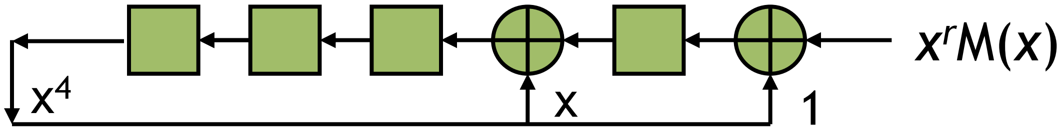
Cálculo da FCS:

Resto [$x^r M(x)/G(x)$] em que *r* é o grau do polinómio G(*x*).

CRC - Implementação

- Bloco a transmitir (M): 1101011011
- Polinómio gerador G(x) = x^4+x+1-> 10011 (grau de G(x) = 4)
- Dados + FCS: 11010110111110

- Registo de deslocamento síncrono:



  - O registo de deslocamento contém *r* bits (nº de bits do FCS);
  - Nº de bits da FCS é igual ao grau de G(x);
  - Existem no máximo *r* ou-exclusivos;
  - A presença ou ausência de um ou-exclusivo corresponde à presença ou à ausência do termo no polinómio G(x).

Série de Problemas nº 4 – Prob. V

Considere uma ligação lógica conforme com a família de protocolos HDLC, em que se adoptou o polinómio $G(x) = x^4 + x^3 + 1$ para o cálculo dos campos CRC das tramas.

a) Calcule a informação para controlo de erros a acrescentar à mensagem: '0111 1011'. Escreva a sequência a transmitir, admita que o primeiro bit a transmitir é o da esquerda.

b) Considere o seguinte padrão erros: '0011 0010 0000'.
   i. Indique o conjunto de bits recebido.
   ii. Justifique, sem efectuar a divisão, se esta trama seria detectada como estando errada (ou não) pelo receptor.
   iii. Considerando que os erros na trama ocorrem de forma independente e que a probabilidade de erro de bit, $P_b = 0{,}25$, indique qual a probabilidade de ocorrência deste padrão de erros.

c) Expresse como uma sequência binária o padrão de erros $E(x)=(x^3+x^2)G(x)$ e calcule a sua probabilidade de ocorrência.

Série de Problemas nº 4 – Prob. VI

Considere que recebeu a seguinte sequência binária:

‘0011 1111 0011 1110 1011 1101 1111 1011’

a) Assinale as bandeiras (*flags*) idênticas ao HDLC (01111110) presentes na sequência recebida.

b) Remova os bits de enchimento (*stuffing*) da mensagem, delimitada pelas bandeiras assinaladas na alínea anterior.

c) Decida sobre a validade da mensagem tendo em conta que o polinómio gerador de código usado foi $G(x) = x^3 + x + 1$.

Nota: Desenhe o registo de deslocamento que implementa o polinómio gerador usado.