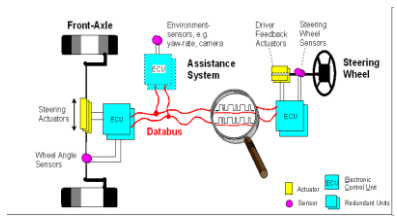
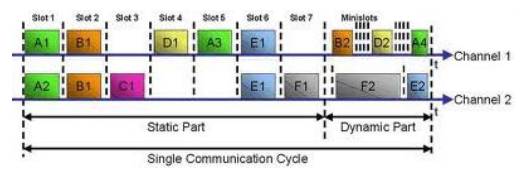
FEUP Universidade do Porto
Faculdade de Engenharia

Redundância Temporal

Introdução

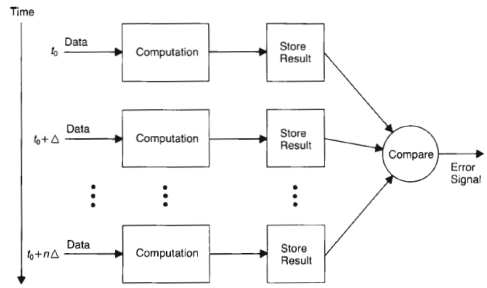
- Redundância temporal:
  - "Executar a acção pretendida em instantes de tempo diferentes"
- Principal objectivo
  - Tolerar falhas transitórias
  - Mas também é possível adapta-la a falhas permanentes
- Exemplo: TDMA (Time Division Multiplexing)





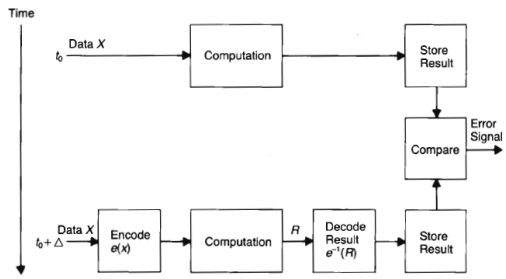
Tolerância a falhas transitórias

- Repetir as acções em diferentes instantes do tempo
- Comparar resultados
- É necessário garantir a consistência dos dados de entrada
- Falhas permanentes podem não ser detectadas (produzem o mesmo resultado)



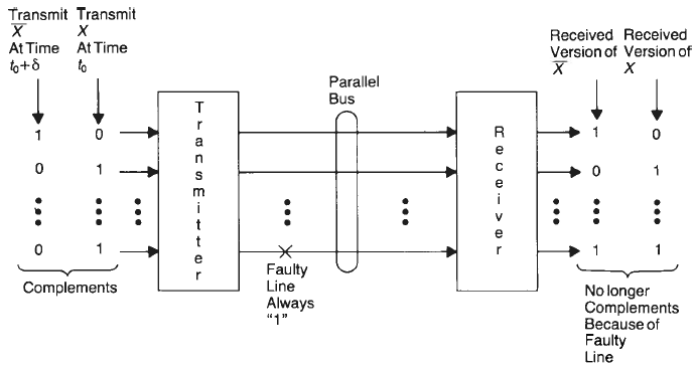
Tolerância a falhas permanentes

- É possível utilizar esta abordagem para detectar falhas permanentes
- Princípio:
  - Executar as acções de forma diferente em instantes diferentes
  - Utilizar uma forma de codificação para executar a 2ª acção
  - Existem várias abordagens



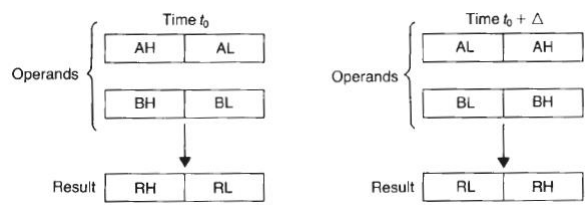
Exemplo: lógica de alternância

- Os dados e o seu complementar são transmitidos alternadamente



Exemplo: Recomputing with swapped operands

- Na 1ª computação os operandos são manipulados na sua forma usual
- Na 2ª computação os 'high e low bytes' são trocados
- Detecção de erros na ALU (Aritmetic Logic Unit)
  - Algumas operações matemáticas (ex. soma)
  - Operações lógicas



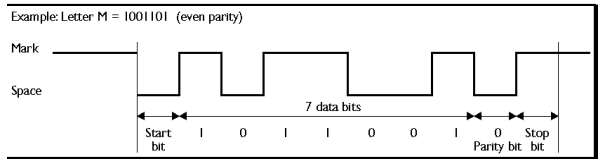
FEUP Universidade do Porto Faculdade de Engenharia

Redundância de Informação

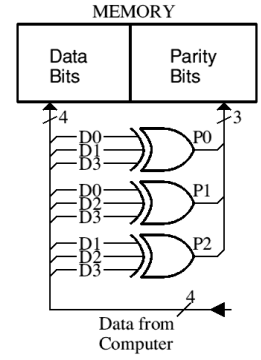
Introdução

- Redundância de informação:
  - "Acrescentar informação redundante aos dados por forma a permitir a detecção, e/ou o mascaramento e eventualmente a tolerância a falhas"

- Exemplos:



Transmissão de dados série



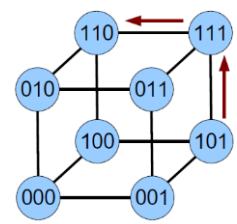
Memória

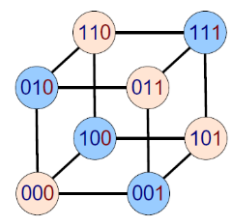
Conceitos

- Alguns conceitos:
  - **Código:** forma de representar a informação utilizando regras bem definidas
    - » Ex: código ASCII : 'A' → 65
    - » Vamos focar a nossa discussão em códigos digitais
  - **Code word** (palavra de código): colecção de símbolos (ou dígitos no caso numérico) utilizados para representar os dados codificados
    - » Ex: BCD : code word de 4 bits para cada dígito
  - **Processo:**
    - » Codificação : Informação → Code word
    - » Descodificação: Code word → Informação

Introdução (cont.)

- Conceitos (cont.):
  - **Distância da Hamming**: número de bits em que duas *code words* diferem
    - » d(101, 110) = 2
    - » d(101, 111) = 1
- A introdução de redundância (bits adicionais) permite definir como válidas apenas um determinado sub-conjunto de todas as palavras de código possíveis...
  - Exemplo: bit de paridade

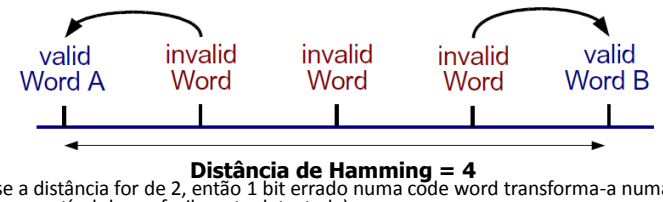




Paridade Par vs. Ímpar

Introdução (cont.)

- Conceitos (cont.):
  - **Distância do código :** distância de Hamming mínima entre duas code words válidas



» Ex. se a distância for de 2, então 1 bit errado numa code word transforma-a numa code word inválida (i.e. susceptível de ser facilmente detectada)

» Para permitir a correcção (para além de apenas a detecção) de erros, torna-se necessário utilizar distâncias de Hamming maiores

Introdução (cont.)

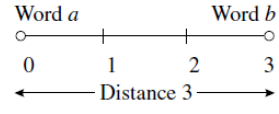
- Conceitos (cont.):
  - Em termos gerais um código pode **corrigir até c erros** (bits errados) e **detectar até d erros** se a sua distância de Hamming (Hd) respeitar a seguinte relação:

$$H_d \geq 2c + d + 1$$

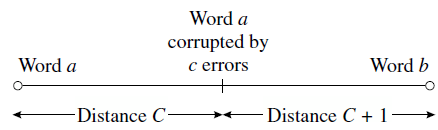
» Para detectar d erros : Hd ≥ d + 1

» Para corrigir c erros : Hd ≥ 2c + 1

» Combinando ambas as expressões obtêm-se a expressão anterior

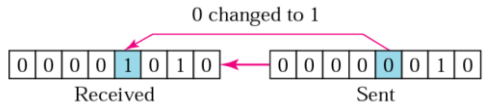


» Ex: códigos com Hd =2 não podem ser utilizados para corrigir erros, mas podem detectar erros simples (1 bit)

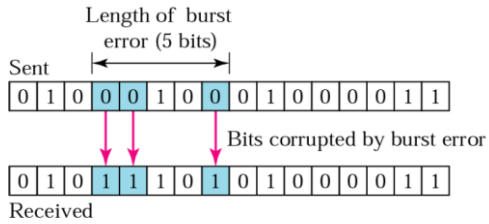


Tipos de erros mais comuns

- Erros simples: afectam um bit, independentes



- Burst de erros: afectam múltiplos bits próximos, não-independentes



Alguns tipos de códigos

- Detecção de erros
  - Os erros produzem *code words* inválidas que podem ser detectadas
  - Tipos:
    - » Paridade
    - » Checksum
    - » Cíclicos (CRCs)
- Correcção de erros
  - Para além da detecção dos erros, as *code words* possuem informação redundante que permite a correcção dos erros
  - Tipos:
    - » Hamming
- ….mas há muitos mais (ver bibliografia)

FEUP Universidade do Porto
Faculdade de Engenharia

Códigos de Detecção de Erros

Códigos de paridade

- Conceito: acrescentar bits que representam a paridade dos dados.
- Caso mais simples: 1 bit
  - Par: número par de 1s
  - Ímpar: número ímpar de 1s
  - Distância de Hamming de 2: apenas um número ímpar de erros (1, 3, ..) é detectado

TABLE 3.3 Odd and even parity codes for BCD data

Decimal digit	BCD	BCD odd parity	BCD even parity
0	0000	0000 **1**	0000 **0**
1	0001	0001 **0**	0001 **1**
2	0010	0010 **0**	0010 **1**
3	0011	0011 **1**	0011 **0**
4	0100	0100 **0**	0100 **1**
5	0101	0101 **1**	0101 **0**
6	0110	0110 **1**	0110 **0**
7	0111	0111 **0**	0111 **1**
8	1000	1000 **0**	1000 **1**
9	1001	1001 **1**	1001 **0**
		↑ Parity bit	↑ Parity bit

Códigos de paridade: exemplo de aplicação

- Detecção de erros em memórias

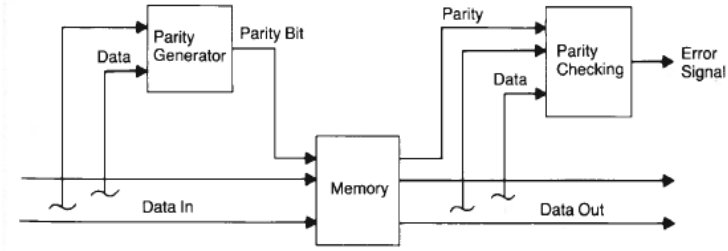
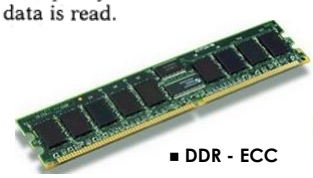


Fig. 3.27 Organization of a memory that uses single-bit parity. The parity bit is generated when data is written to memory and checked when data is read.



■ DDR - ECC

Códigos de paridade: geradores / detectores

- O mesmo circuito serve as duas funcionalidades

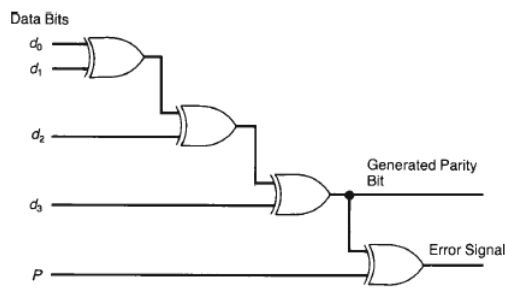
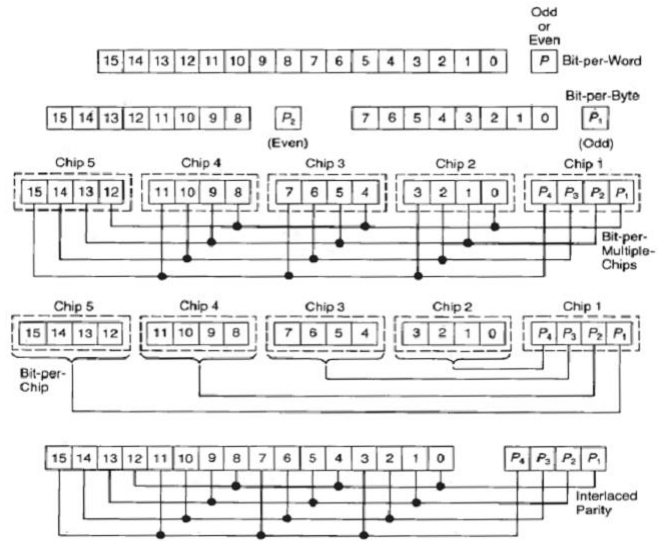


Fig. 3.28 A 4-bit parity generation and checking circuit for even parity.

Códigos de paridade: diferentes abordagens

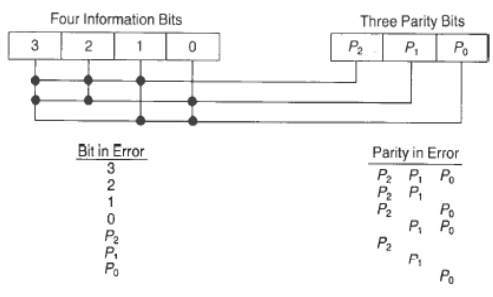
- **Paridade por word**
  - Impossibilidade de detectar situações de tudo 1s ou 0s
- **Paridade por byte**
  - Detectar até 2 erros
- **Paridade por múltiplos chips**
  - Detectar múltiplos erros
  - Detectar um chip totalmente avariado
- **Paridade por chip**
  - Detectar qual o chip que contém o erro
  - Não detecta um chip totalmente avariado
- **Paridade entrelaçada**
  - Independente da organização da memória
  - Detectar erros em bits adjacentes

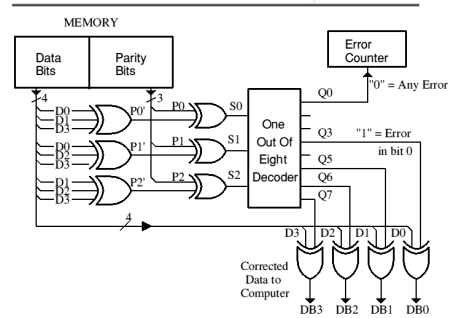


Códigos de paridade: diferentes abordagens (2)

- Paridade com sobreposição:
  - Organizar 'grupos de paridade'
  - Erros são detectados e localizados (à custa de um overhead adicional)
  - Possibilidade de corrigir o bit errado

Number of information bits	Number of parity bits	Redundancy percentage
2	3	150.0%
4	3	75.0
6	4	66.7
8	4	50.0
10	4	40.0
12	5	41.7
16	5	31.25
24	5	20.8
32	6	18.75
64	7	10.9





Códigos m-of-n

- Todas as code words têm comprimento **n** e contêm exactamente **m** 1s (**m** bits concatenados aos dados)
- Distância de Hamming de 2 (1 erro implica que o número de 1s: +1 ou -1)
  - Mas podem detectar múltiplos erros em situações especiais (ex: vários 1 passarem a 0)
- Processo de codificação / descodificação podem ser complexos
- Overhead elevado

TABLE 3.5 3-of-6 code for representing three bits of information

Original information	3-of-6 code	
000	000	111
001	001	110
010	010	101
011	011	100
100	100	011
101	101	010
110	110	001
111	111	000
	Original information	Appended information

Códigos de duplicação

- Duplicar a informação (overhead elevado)
- Informação complementada (duplicação complementada)
- Existem outras variantes (ex: swap and compare)

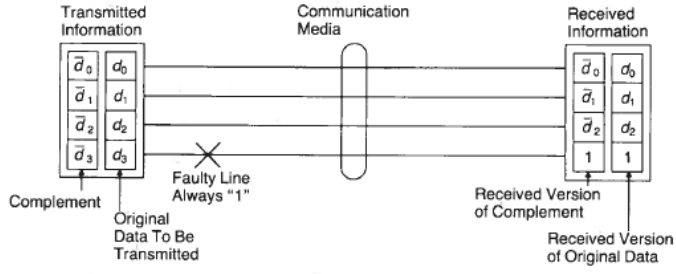
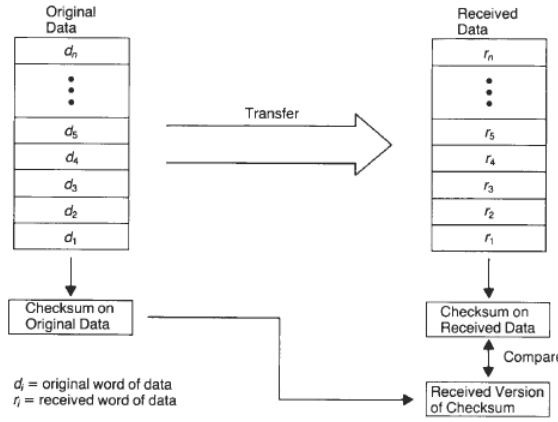


Fig. 3.33 Example of complemented duplication for error detection in a communication system.

Checksums

- **Checksum = soma**

- Muito utilizado em sistemas de comunicação
- Simples de implementar
- Os dados a transmitir são 'somados' e o resultado (checksum) é transmitido juntamente com os dados
- O receptor calcula o checksum dos dados recebidos e compara-a com o checksum recebido. Se diferirem há erros.

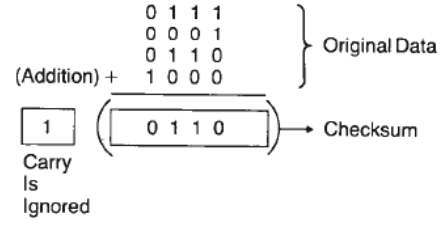


Checksums (discussão)

- O checksum pode ter a mesma 'dimensão' dos dados (ex: byte) ou superior
- A 'soma' pode realizar-se de várias formas:
  - Módulo 2 (XOR)
  - Precisão simples (soma binária : **n** bits → **n** bits)
  - Dupla precisão (**n** bits → **2n** bits)
  - Outras (ver bibliografia)

Exemplo:

10010010
01010010
11110000
00001001
10010010
10101011 acrescentado à mensagem



Checksum : precisão simples

- A capacidade de detecção depende do tipo de checksum utilizado

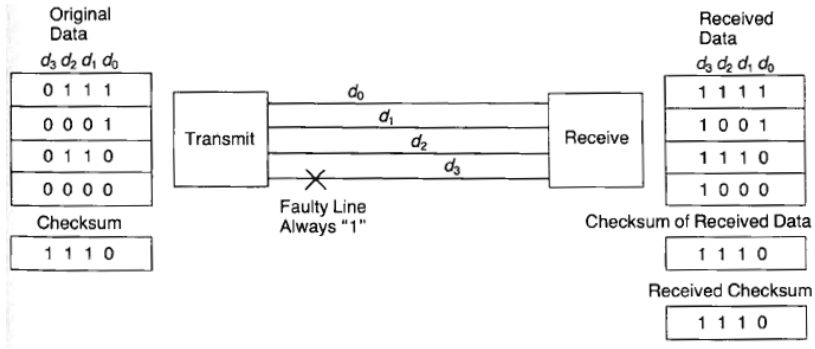


Fig. 3.37 The single-precision checksum is unable to detect certain types of errors. The received checksum and the checksum of the received data are equal, so no error is detected.

Códigos cíclicos

- Cyclic Redundancy Codes (CRC)
  - Os bits de paridade não são adequados à detecção de burts de erros, ou implicam overheards elevados para a sua detecção de
- Os CRC são baseados em cálculos matemáticos realizados sobre os dados (mensagem): divisão polinomial, cujo resto tem que ser 0
- Princípio:
  - Mensagem **M** com **k** bits
  - O emissor gera uma sequência de **n** bits denominada **FCS** (*Frame Check Sequence*), por forma que a concatenação de **M** com o **FCS** (**k+n** bits) seja divisível **G** (divisível = resto 0)
  - **G** é denominado polinómio gerador (**n+1** bits)
  - A mensagem transmitida **T** é obtida como: multiplicando **M** por **2n** (shift 2n x) e adicionando (módulo-2 = XOR) o **FCS**

$$T = 2^n \cdot M + F$$

CRC (discussão)

- Dividindo **2^nM** por **G** obtêm-se um quociente **Q** e um resto **R**

$$\frac{2^n \cdot M}{G} = Q + \frac{R}{G}$$

- Utilizando como **R** o valor do **FCS** temos

$$T = 2^n \cdot M + R$$

- No receptor, realizando a divisão por **G** temos

$$\frac{T}{G} = \frac{2^n \cdot M + R}{G} = Q + \frac{R}{G} + \frac{R}{G} = Q$$

- Isto é, o resto é 0 !

Discussão (2)

- Se o resto da divisão for 0 então não há erro na transmissão:
  - **Prova**: assumir que a mensagem recebida é a 'soma' (XOR) da mensagem original **T** com um erro **E** (ex. 0000010000)

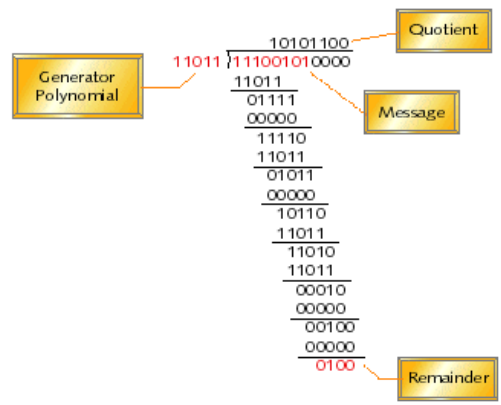
$$\frac{T+E}{G} = \frac{T}{G} + \frac{E}{G}$$

  - Dado que **E/G** não dá resto zero (não devia dar…), é porque ocorreu um erro na transmissão.
- Resumo:
  - Emissor: dividir **2^nM** por **G** obter o resto (**FCS**), adicionar a **M** e obter T
  - Receptor : dividir **T** por **G** (se resto = 0 não há erros)
- Pergunta : como escolher o **G** (polinómio gerador)?

Discussão (3)

- Uma unidade de dados com **n** bits de informação é representada por um polinómio **D(x)** de grau **n-1** : o valor de cada bit é o coeficiente de um termo do polinómio
  - **D = 110011 : D(x) = $X^5 + X^4 + X + 1$**

- Divisão (exemplo):
  - M(x) = 11100101
  - G(x) = 11011
  - Aritmética de módulo-2 (XOR)



Discussão (5)

- Os CRC detectam :
  - Todos os erros simples, desde que **G(x)** tenha coeficientes não nulos
  - Todos os erros duplos, se **P(x)** contiver um factor com pelo menos três termos
  - Qualquer número ímpar de erros, se **P(x)** contiver o factor (**x + 1**)
  - Qualquer burst de erros com comprimento inferior ou igual a **n** bits, e a maior parte de bursts de erros com comprimento superior a **n** bits, se **P(x)** tiver grau **n**
  - Para bursts de erros > **n** a probabilidade do erro não ser detectado é ~ **$1/2^n$**

- Exemplos de polinómios:
  - CRC-32 : $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$ (Ethernet)

FEUP Universidade do Porto
Faculdade de Engenharia

Códigos de Correcção de Erros

Códigos de Hamming

- Derivados dos códigos de sobreposição de paridade:
  - Os dados (bits) a transmitir são agrupados e para cada grupo é obtida a respectiva paridade
  - Os grupos não são mutuamente exclusivos (daí a sobreposição)
  - Para **k** bits de dados utiliza **c** bits de paridade, respeitando a seguinte condição:

$$2^c \geq c + k + 1$$

  - A code word tem dimensão **n = c + k**
- Princípio:
  - A ocorrência de erros 'gera' uma *code word* única que permite identificar (dentro de certas condições) os bits errados e assim permitir a sua correcção
  - Esta code word é denominada **síndrome**

Códigos de Hamming (discussão)

- Exemplo
  - O síndroma 'indica' qual o bit errado
    - » '0' sem erros
  - A organização dos bits segue uma estrutura específica

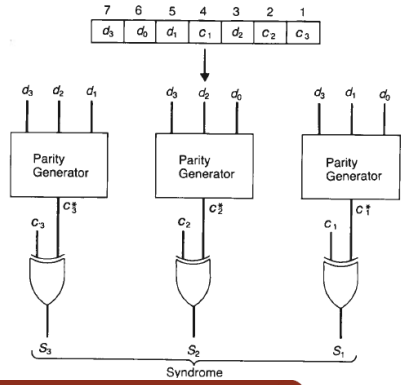


TABLE 3.17 Check bits affected by single data bit errors

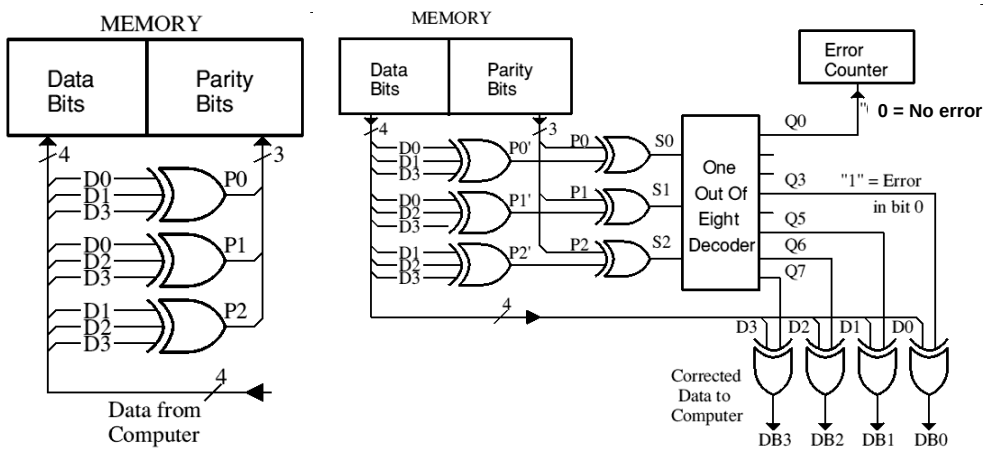
Erroneous bit	Check bits affected
d_0	c_1, c_2
d_1	c_1, c_3
d_2	c_2, c_3
d_3	c_1, c_2, c_3
c_1	c_1
c_2	c_2
c_3	c_3

TABLE 3.18 Resulting syndromes for each possible single bit error

Erroneous bit	Syndromes
d_0	110
d_1	101
d_2	011
d_3	111
c_1	100
c_2	010
c_3	001

Códigos de Hamming (discussão)

- Detecção e correcção de erros em memórias



Bibliografia

- **Principal**:
  - "Design and Analysis of Fault-Tolerant Digital Systems", Barry Johnson (nos conteúdos):
    - » Capitulo 3 : Design Techniques to Achieve Fault Tolerance

- **Complementar**
  - "Reliability of Computer Systems and Networks – Fault Tolerance, Analysis and Design", Martin Shooman, Wiley, (na biblioteca):
    - » Capitulo 2 : Coding Techniques