Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Unidade Acadêmica de Sistemas e Computação
Curso de Bacharelado em Ciência da Computação

Organização e Arquitetura de Computadores

Circuitos Lógicos Combinacionais (Detecção e Correção de Erros)

Profa. Joseana Macêdo Fechine Régis de Araújo
joseana@computacao.ufcg.edu.br

Carga Horária: 60 horas

Tópicos

Circuitos Lógicos Combinacionais

- Gerador/Verificador de Paridade
- Outros circuitos (Checksum, CRC, ...)

Projeto de Circuitos Combinacionais

Circuito Gerador/Verificador de Paridade

- Utilizado para detectar erro em transmissão digital.
- Este processo pode ser vulnerável se houver mais do que um erro, permitindo assim que este passe até o destino sem ser identificado.
- Usado em muitas aplicações de hardware (em que uma operação pode ser repetida em caso de dificuldade, ou quando é útil a simples detecção de erros).

Projeto de Circuitos Combinacionais

Bit de Paridade

- Bit extra anexado ao conjunto de bits para informar a sua paridade.
- O bit de paridade pode ser 0 ou 1, dependendo do número de 1´s contido no conjunto de bits do código (par ou ímpar).
- **Paridade Par**: o bit anexado serve para tornar o número total de bits “1” **par** (Ex.: 01001 -> **0**01001).
- **Paridade Ímpar**: o bit anexado serve para tornar o número total de bits “1” **ímpar** (Ex.: 01001-> **1**01001).

Projeto de Circuitos Combinacionais

Código de Paridade

Código de Paridade Par: é um código formado pela adição de um bit de controle à informação original, de forma a produzir uma *codeword* (palavra-código) com um número par de 1, ou seja:

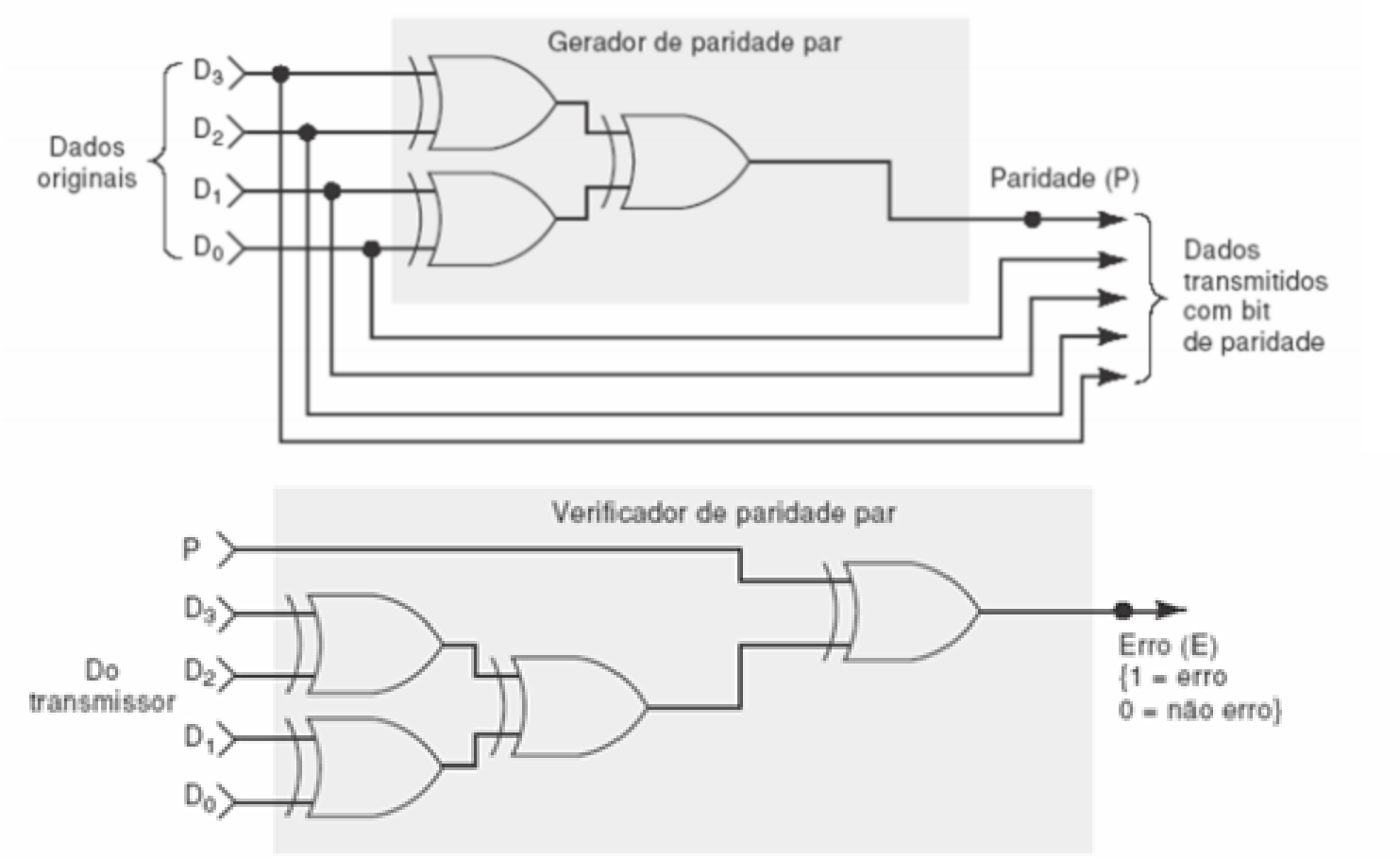
$$C = b_0 \oplus b_1 \oplus \ldots b_k$$

Código de Paridade Ímpar: é um código formado pela adição de um bit de controle à informação original, de forma a produzir uma *codeword* (palavra-código) com um número ímpar de 1, ou seja:

$$C = 1 \oplus b_0 \oplus b_1 \oplus \ldots b_k \quad \text{ou} \quad C = \overline{b_0 \oplus b_1 \oplus \ldots b_k}$$

Projeto de Circuitos Combinacionais

Exemplo: Geração e Verificação de Paridade Par (4 bits)



Projeto de Circuitos Combinacionais

Código de Paridade

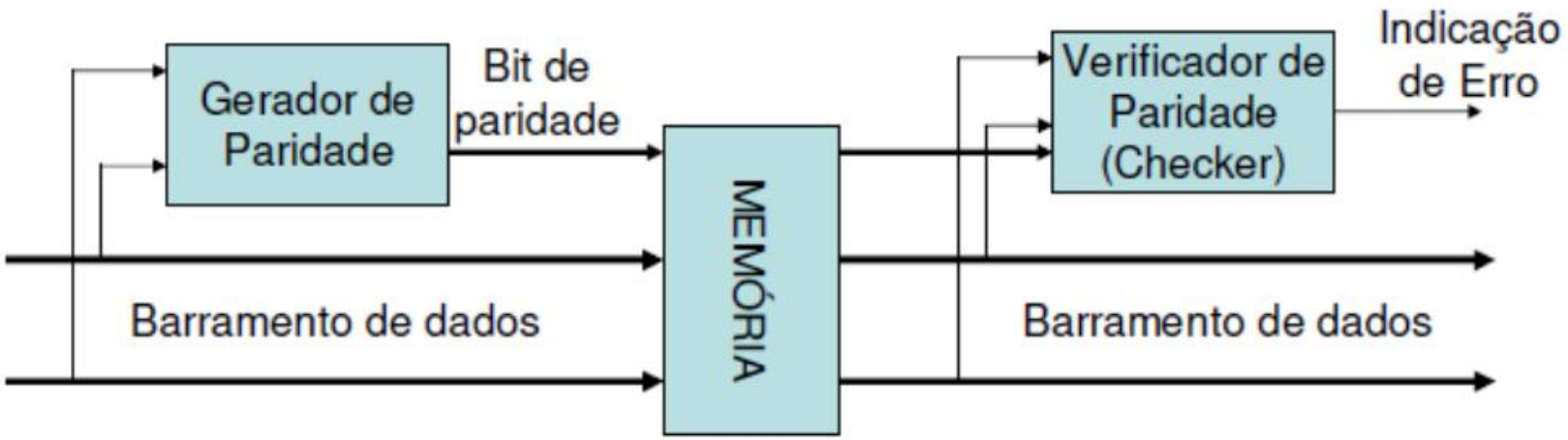
Código de Detecção de Erros

Exemplo Simples:

- Inclusão de 1 bit de paridade (0 - par e 1 - ímpar) aos bits de dados da palavra-código.
- A ocorrência de 1 único erro produz palavra-código errada.
- **Erro só é detectado, e não corrigido**.
- Programa cancela o processamento para não gerar resultados errados.

Projeto de Circuitos Combinacionais

Código de Paridade - Exemplo



Projeto de Circuitos Combinacionais

Outras estratégias para detecção de erros

- ***Checksum*** - Consiste na transmissão de todas as palavras juntamente com o resultado da sua soma binária.
- **CRC** (***Cyclic Redundancy Check***).
- **Códigos de Hamming** - Detecção e Correção de Erros.

Projeto de Circuitos Combinacionais

Checksum

- Princípio básico
  - Somar todas as palavras que são transmitidas e depois transmitir o resultado daquela soma.
  - Se quaisquer dos dados transmitidos, incluindo o *checksum*, sofrerem algum erro, então, no receptor, o resultado da operação soma não será correto.
  - Existem variações na forma de implementação (uso do complemento de 1 ou do complemento de 2).

Projeto de Circuitos Combinacionais

Checksum (Exemplo)

- Transmitir todas as palavras juntamente com o resultado da sua **soma binária invertida**.
- Incluir o bit de transporte (vai-um), exceto o último.
- Realizar a inversão do valor dos bits (**complemento de 1**) do *checksum*.
- Transmitir as palavras e o *checksum* (invertido).

A	B	SOMA	VAI-UM
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Projeto de Circuitos Combinacionais

Checksum - Exemplo

- *Checksum* de 2 palavras de 8 bits
- **Dados iniciais**: 00111101 e 00001101
- ***Checksum***: 01001010
- ***Checksum* invertido (complemento de 1)**: 10110101
- Dados enviados:
  - 00111101
  - 00001101
  - 10110101 (checksum invertido)
- No receptor: as palavras são novamente somadas e comparadas com o *checksum* enviado
  - Se qualquer um dos dados transmitidos, incluindo o *checksum*, sofrerem algum erro, então, a soma do novo *checksum* com o *checksum* enviado, será diferente de 1.

Projeto de Circuitos Combinacionais

Checksum - Exemplo

- *Checksum* de 2 palavras de 8 bits
- **Dados iniciais**: 00111101 e 00001101
- ***Checksum***: 01001010 e ***Checksum* invertido**: 10110101
- Dados enviados:
  - 00111101
  - 00001101
  - 10110101 (*checksum* – invertido)

0 1 0 0 1 0 1 0 (novo *checksum* - das palavras iniciais recebidas)

1 0 1 1 0 1 0 1 (*checksum* enviado)

1 1 1 1 1 1 1 1 (resultado da soma - sem erro)

Projeto de Circuitos Combinacionais

Checksum – Exemplo (Erro)

Dados iniciais: 00111101 e 00001101

10110101 (*checksum* – invertido)

0 0 1 1 **0 0** 0 1 (erros indicados em vermelho)

0 0 0 0 1 1 0 1

0 0 1 1 1 1 1 0 (novo *checksum*)

1 0 1 1 0 1 0 1 (*checksum* enviado)

1 1 1 1 **0 0** 1 1 ≠ 1 1 1 1 1 1 1 1 **(resultado da soma - com erro)**

- Valor recebido incorretamente, **com erro no 3º ou 4º bit** (de qualquer uma das palavras enviadas, incluindo o *checksum*)

Projeto de Circuitos Combinacionais

Checksum (outro forma - exemplo)

- Para 4 bytes de dados: 0x34, 0x51, 0x4E, 0x63.
  - 1: Somar todos os bytes: 0x136.
  - 2: Retirar o último vai-um (*carry nibble*): 0x36.
  - 3: **Calcular o complemento de dois** de 0x36: 0xCA. Esse é o byte de *checksum*.
  - 4: Testar o byte *checksum*: adicioná-lo ao grupo original de bytes (0x36 + 0xCA). O resultado será: 0x100.
  - 5: Retirar o último vai-um: 0x00. **O resultado 0x00** (0 em decimal) indica que não houve erro (mas, um erro indetectável pode ter ocorrido).

Projeto de Circuitos Combinacionais

Códigos para Correção de Erros

- Exemplos: código de Hamming, código Reed-Salomon, BCH, etc.

Um código simples para correção de erros

- Faz uso de bits de paridade "cruzados“ e é capaz de corrigir quaisquer erros de 1 bit.

Projeto de Circuitos Combinacionais

Exemplo: Uma mensagem de 16 bits dispostos como um quadrado 4x4.

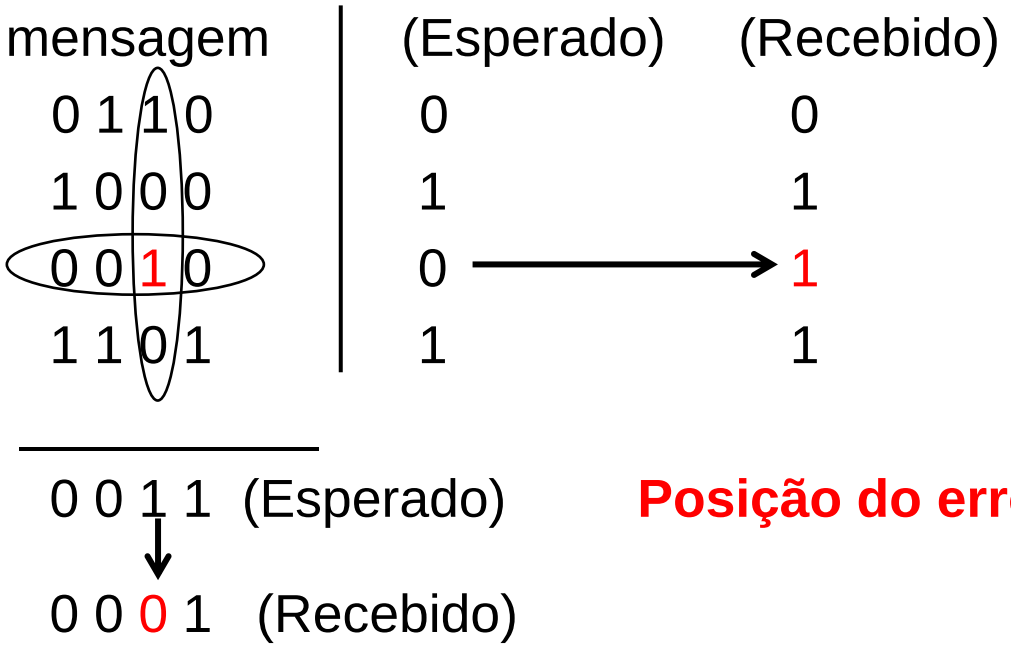
mensagem	paridade vertical (par)
0 1 1 0	0
1 0 0 0	1
0 0 0 0	0
1 1 0 1	1

0 0 1 1

paridade horizontal (par)

Projeto de Circuitos Combinacionais

Exemplo: A paridade cruzada pode corrigir qualquer erro de um bit, pois os erros de paridade dão as coordenadas do bit errado.



Posição do erro: linha 3, coluna 3

Projeto de Circuitos Combinacionais

Exemplo: Se a paridade indicar erro em apenas uma dimensão, a corrupção ocorreu no próprio bit de paridade e não na mensagem.

mensagem	
0 1 1 0	0
1 0 0 0	1
0 0 0 0	0
1 1 0 1	1
0 0 0 1	

↑ paridade errada

Este código simples não pode resolver erros de dois bits ou mais, pois a posição dos erros torna-se ambígua.

Projeto de Circuitos Combinacionais

Códigos de Hamming

- O algoritmo de Hamming adiciona bits extras nas posições que são potência de 2 (Ex.: palavra de 32 bits: 1, 2, 4 ,8, 16, 32).
- O tamanho da palavra (32 bits) somado ao tamanho dos bits extras (6 bits) formam o tamanho da palavra-código (38 bits).
- A distância de Hamming é o número de bits diferentes (ou número de erros) entre duas palavras-código (válidas).
- Operação realizada no momento de admissão da palavra com a inclusão de bits extras.
- Processo inverso realizado no acesso à palavra com a remoção de bits extras para obtenção da palavra original.
- Pelo menos duas paridades precisam estar erradas para que um bit da mensagem original esteja errado. Se apenas uma paridade estiver errada, isto significa que é a própria paridade que foi corrompida, não a mensagem original.

Projeto de Circuitos Combinacionais

Exemplo

- Considere uma mensagem de **4 bits**.
- Os bits de paridade são inseridos nas posições que são potências de 2 (1, 2, 4) assumindo que a contagem das posições começa em 1.
- Os bits de paridade (p1,.., p3) são misturados aos bits da mensagem original (m1, .., m4).

Posição do bit	1	2	3	4	5	6	7
Bits	p1	p2	m1	p3	m2	m3	m4

Tamanho da palavra-código: 7 bits

Projeto de Circuitos Combinacionais

Exemplo: mensagem (4 bits): 0101 (5 decimal)

Número de bits extra (paridade): 3 (potências de 2: 1, 2, 4)

Posição do bit	1	2	3	4	5	6	7
Bits	p1	p2	m1	p3	m2	m3	m4
	0	**1**	0	**0**	1	0	1

- Índices do bits de dados (mensagem)
  - m1 (posição 3) -> 2 + 1 -> p2 + p1
  - m2 (posição 5) -> 4 + 1 -> p3 + p1
  - m3 (posição 6) -> 4 + 2 -> p3 + p2
  - m4 (posição 7) -> 4 + 2 + 1 -> p3 + p2 + p1

- **Geração dos bits de Paridade**
  - p1 = m1 ^ m2 ^ m4 **p1 = 0 ^ 1 ^ 1 = 0**
  - p2 = m1 ^ m3 ^ m4 **p2 = 0 ^ 0 ^ 1 = 1**
  - p3 = m2 ^ m3 ^ m4 **p3 = 1 ^ 0 ^ 1 = 0**

Projeto de Circuitos Combinacionais

Exemplo: mensagem (4 bits): 0101 (5 decimal)

Número de bits extra (paridade): 3 (potências de 2: 1, 2, 4)

Posição do bit	1	2	3	4	5	6	7
Bits	p1	p2	m1	p3	m2	m3	m4
	0	**1**	0	**0**	1	0	1

- **Ex.: Palavra : 0100101 -> 0101**
- **Verificação de Erro**
  - v1 = p1 ^ (m1 ^ m2 ^ m4) **v1 = 0 ^ 0 ^ 1 ^ 1 = 0** **Código está correto.**
  - v2 = p2 ^ (m1 ^ m3 ^ m4) **v2 = 1 ^ 0 ^ 0 ^ 1 = 0**
  - v3 = p3 ^ (m2 ^ m3 ^ m4) **v3 = 0 ^ 1 ^ 0 ^ 1 = 0**
- **Ex.: Palavra : 0100111 -> 0100101**
- **Verificação de Erro**
  - v1 = p1 ^ m1 ^ m2 ^ m4 **v1 = 0 ^ 0 ^ 1 ^ 1 = 0** **Código está errado**
  - v2 = p2 ^ m1 ^ m3 ^ m4 **v2 = 1 ^ 0 ^ 1 ^ 1 = 1** **Bit errado: 6 (110)**
  - v3 = p3 ^ m2 ^ m3 ^ m4 **v3 = 0 ^ 1 ^ 1 ^ 1 = 1** **Correção: 0100101**

Projeto de Circuitos Combinacionais

Códigos de Hamming

- Para detectar *d* erros é necessário um código com distância mínima igual a $d_{mín}$= *d*+1.
- Para corrigir *d* erros é necessário um código com distância mínima igual a $d_{mín}$= 2*d*+1.
- A Distância Mínima ($d_{mín}$) de um código é a menor Distância de Hamming encontrada, considerando todas as combinações válidas de palavras do código.

Projeto de Circuitos Combinacionais

Detecção e Correção de Erros

- Detecção e correção de erros (EDC - *Error Detection and Correction bits* - redundância) são técnicas muito importantes no mundo atual.

- Os meios de comunicação atuais dependem de EDC para funcionar e a computação depende da detecção de erros para se confiável.

- **O que aconteceria se uma operação crítica pudesse ser corrompida no caminho?**