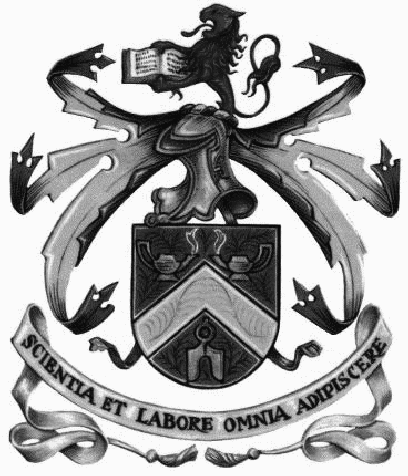
**Universidade de Trás-os-Montes e Alto Douro Departamento de Engenharias**

Sistemas Digitais

1º Ano de Engenharia Informática



Trabalho Prático n.º 3

*OU–Exclusivo*

Grupo



Tiago José Ferreira Fernandes n.º 73701

n.º

n.º

Turma 4

FMGG / 2008

## Objectivos

* Investigar as propriedades lógicas da função **OU–Exclusivo** (**XOR**)
* Implementar várias funções lógicas usando portas XOR.

## Referências

* TAUB, Herbert, “Circuitos Digitais e Microprocessadores”, McGraw–Hill
* Texas Instruments online [<http://www.ti.com/>]

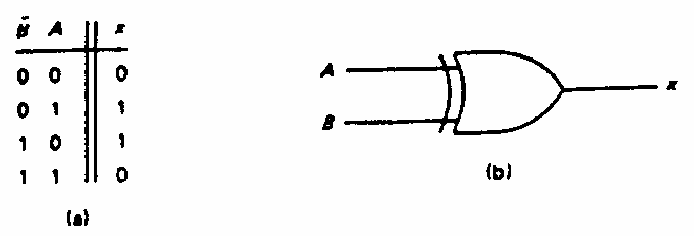
## Material

* Placa RH21
* CI 74LS00 — NAND
* CI 74LS04 — NOT
* CI 74LS08 — AND
* CI 74LS32 — OR
* CI 74LS86 — XOR

# A função OU–Exclusivo

A função OU–Exclusivo (XOR) é uma função interessante e útil. Como o nome indica, tem alguma semelhança com a função OU (OR) já estudada, mas é uma operação distinta. A porta ou–exclusivo pode ser usada para implementar operações lógicas, tais como a **verificação de pa- ridade**, **conversão de numeração binária para *Gray***, a adição binária de números, e muitas ou- tras.

A tabela de verdade e o símbolo lógico da função OU–Exclusivo (**XOR** nas próximas referên- cias) são apresentadas na figura 1.

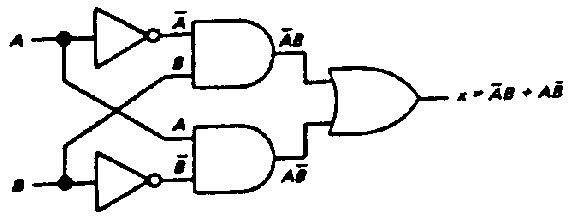


**Figura 1.** Porta OU–Exclusivo. (a) Tabela de verdade (b) Símbolo lógico

# Portas XOR

O modo mais directo de implementar a função XOR é ligar dois inversores, duas portas AND e uma porta OR como mostrado na figura 2. A função pode ser escrita como:

*x*  *A*  *B*  *AB*  *AB*



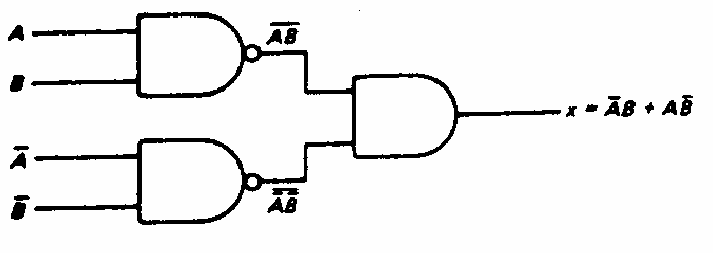
**Figura 2.** Função XOR com portas NOT, AND e OR.

* 1. Monte o circuito da figura 2. Verifique, completando a tabela de verdade, que este cir- cuito executa a função XOR.

|  |  |  |
| --- | --- | --- |
| *A* | *B* | *x*  *AB*  *AB* |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Um segundo circuito lógico que executa a operação XOR é mostrado na figura 3. A sua função lógica é a seguinte:

*x*  *AB*.*A*.*B*



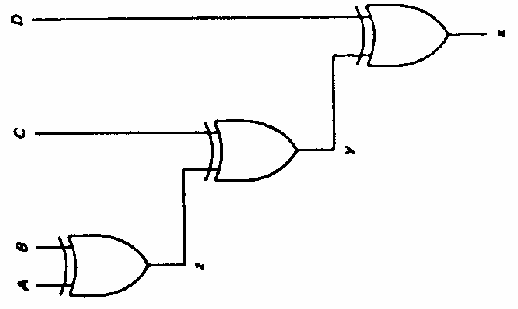
**Figura 3.** Função XOR com portas AND e NAND.

* 1. Monte o circuito da figura 3 e complete a tabela de verdade apresentada na página seguinte para verificar que o circuito implementa a função XOR.

|  |  |  |
| --- | --- | --- |
| *A* | *B* | *x*  *AB*.*A*.*B* |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Como seria de esperar, a função XOR está disponível em CI. Por exemplo, o **74LS86** é um circuito integrado que oferece quatro portas XOR de duas entradas numa embalagem de 14 pi- nos.

# Circuito de verificação de paridade

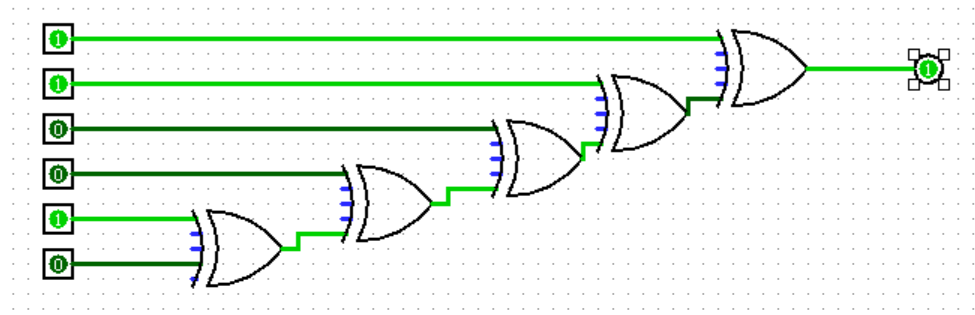
A **paridade** de um número binário refere-se ao número de **1**’s que essa palavra binária con- tém. Assim, se existir um número par de **1**’s, a paridade é **par**, caso contrário a paridade é **ím- par**. Por exemplo, a palavra binária 1001110 tem quatro **1**’s, logo tem paridade par. O circuito da figura 4 apresenta um meio de verificar a paridade de uma palavra de quatro bits, ABCD.

**Figura 4.** Geração do bit de paridade.

* 1. Usando o ’86, construa o circuito de teste da figura 4. Verifique o funcionamento correcto do circuito aplicando quatro palavras de paridade par e quatro palavras de paridade ímpar, e anote a saída *x* (nenhuma das entradas deve ser constante em todas as oito palavras testadas). Registe os resultados na tabela seguinte:

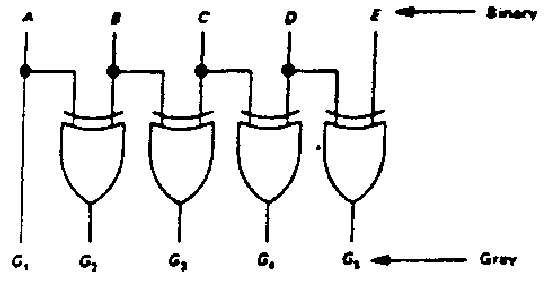
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *A* | *B* | *C* | *D* | *X* |
| 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |

* 1. Desenhe o diagrama lógico de um circuito de teste de paridade para palavras de 6 bits, usando portas XOR. Indique a saída de cada porta quando a entrada é a palavra 110010.



# Conversão Binário–Gray

Outra aplicação para portas XOR é o circuito lógico para converter uma palavra em código binário para o seu equivalente em código Gray, como se mostra na figura 5.



**Figura 5.** Conversor Binário–Gray.

* 1. Use o ’86 para implementar o conversor de binário para Gray da figura 5. Verifique o funcionamento correcto do circuito aplicando as entradas binárias (ABCDE) correspondentes aos valores decimais indicados no quadro abaixo e registando as saídas Gray (G’s).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Dec.* | *A* | *B* | *C* | *D* | *E* | *G*1 | *G*2 | *G*3 | *G*4 | *G*5 |
| 15 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 16 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 17 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 19 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 20 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

* 1. Desenhe (**em anexo**) o diagrama lógico de um conversor Binário–Gray para palavras de 9 bits. Indique a saída de cada porta se a entrada for 110001010.

