# Circuitos Digitais (116351) - 4º Experimento

## CIRCUITOS COMBINACIONAIS: COMPARADOR DE PALAVRAS

**OBJETIVO:** Esta experiência visa a descrição e projeto de um comparador de palavras binárias, usando-se as técnicas de síntese de circuitos combinacionais já vistas: tabela da verdade, simplificação por mapa de Karnaugh e implementação da função booleana simplificada com portas lógicas.

### 1. INTRODUÇÃO TEÓRICA

#### 1.1. GENERALIDADES

O projeto de circuito combinacional envolve quase sempre 5 passos, a saber:

a) Descrição do sistema;
b) Elaboração da tabela da verdade;
c) Obtenção das funções booleanas a partir da tabela da verdade;
d) Simplificação das funções booleanas obtidas (métodos de minimização); e
e) Elaboração do diagrama lógico a partir das funções booleanas simplificadas.

O **comprimento** de uma palavra binária é o número de *bits* que compõem a palavra. Por exemplo, a palavra binária 01101 possui 5 *bits*.

Será visto nesta introdução apenas a descrição do sistema. As outras etapas do projeto serão cumpridas pelo aluno na parte experimental.

#### 1.2. DESCRIÇÃO DO SISTEMA

Um comparador de palavras, como seu próprio nome diz, compara duas palavras de *n bits* cada uma. Quando as duas palavras forem iguais, a saída será 1; caso contrário a saída deverá ser 0.

O diagrama de blocos de um comparador de palavras de 4 *bits* é mostrado na **Figura 1**.

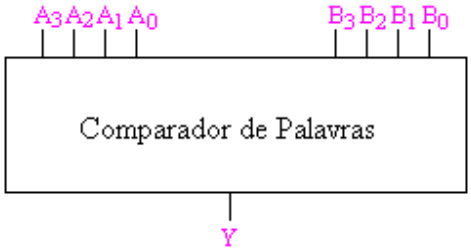


**Figura 1 – Comparador de palavras de 4 *bits***

Se as palavras A = $A_3A_2A_1A_0$ e B = $B_3B_2B_1B_0$ forem iguais, a saída *Y* deverá ser 1. Se forem diferentes, a saída *Y* deverá ser 0.

Um comparador de palavras de 4 *bits* pode ser facilmente implementado com portas XNOR's como mostra a **Figura 2**.

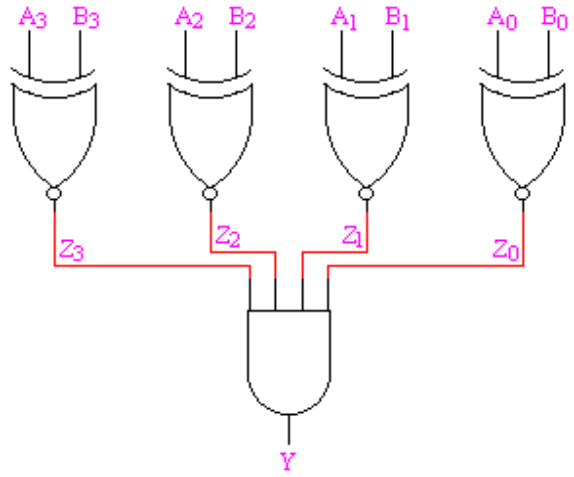


**Figura 2 – Implementação do comparador da Figura 1**

Na **Figura 2**, a saída será 1 somente quando A = B.

## 2. PARTE EXPERIMENTAL

2.1. Projetar e montar um comparador de palavras de 3 *bits*, usando apenas portas NAND de duas entradas.

a) Complete a tabela da verdade abaixo.

| **Entradas** | | **Saída** |
|---|---|---|
| $\mathbf{A_i}$ | $\mathbf{B_i}$ | $\mathbf{Z_i}$ |
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

b) Obtenha a função booleana $Z_i$.
c) Minimize a função obtida no item **b**.
d) Faça um diagrama lógico parcial (comparação do par de bits ($A_i$, $B_i$)).
e) Implemente este diagrama parcial e verifique a tabela da verdade do item **a**.
f) Faça um diagrama lógico total.
g) Implemente este diagrama lógico total.
h) Apresente os resultados obtidos em forma de tabela da verdade e faça comentários.

2.2. Projetar um comparador de duas palavras de 2 *bits*, com 3 saídas, tal que $Y_1 = 1 \Leftrightarrow A > B$, $Y_2 = 1 \Leftrightarrow A = B$ e $Y_3 = 1 \Leftrightarrow A < B$. Usar portas AND, OR, NOT e XOR.

$A = A_1A_0$
$B = B_1B_0$

Sugestão: Divida o problema em 2 comparadores de 2 *bits*. A primeira comparação é feita com os *bits* mais significativos. Se forem iguais é feita a segunda comparação.

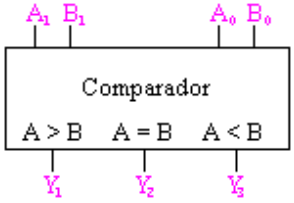


**Figura 3 – Comparado com 3 saídas**

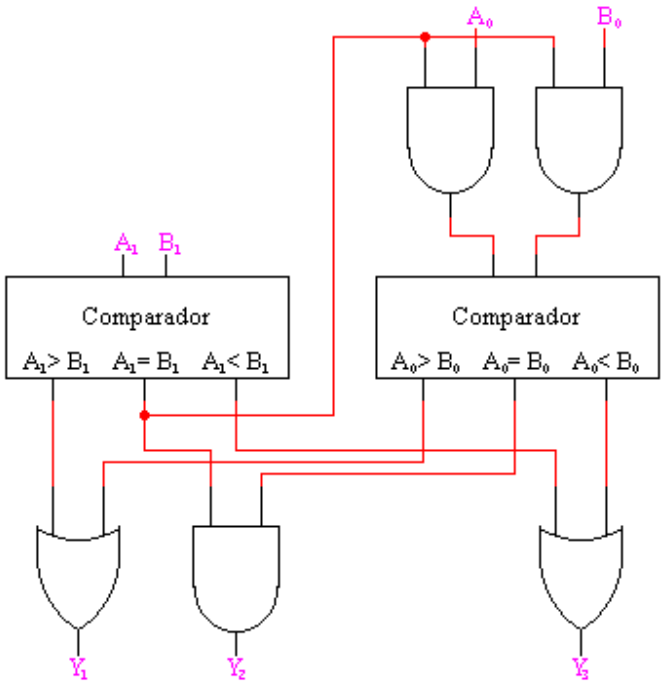


**Figura 4 – Implementação da Figura 3**

a) Elabore uma tabela da verdade parcial.
b) Obtenha as funções booleanas parciais.
c) Minimize as expressões obtidas no item **b**.
d) Faça um diagrama lógico parcial (comparação do *bit* mais significativo).
e) Implemente este diagrama lógico parcial e verifique a tabela da verdade do item **a**.
f) De acordo com a **Figura 4**, faça o diagrama lógico total.
g) Implemente este diagrama lógico total.
h) Apresente os resultados obtidos em forma de tabela da verdade e faça comentários.

## 3. SUMÁRIO

Um comparador de duas palavras de mesmo comprimento é estudado. Se as duas palavras forem iguais, a saída será 1; caso contrário a saída será 0. É apresentado, também, um comparador de 3 saídas $Y_1$, $Y_2$ e $Y_3$ que será 1, respectivamente, quando A > B, A = B e A < B.

## 4. EQUIPAMENTOS E MATERIAL

- software Quartus II
- FPGA Altera Cyclone II.

## 5. TESTE DE AUTO-AVALIAÇÃO

1. Implemente um comparador de palavras de 4 *bits* em que a saída seja 1 somente quando A = B. Use 4 portas XOR e obtenha um diagrama com um total de 5 portas, como o da **Figura 2**. A porta de saída será uma:
   a) NAND de 4 entradas.
   b) NOR de 4 entradas.
   c) AND de 4 entradas.
   d) OR de 4 entradas.

2. Implemente um comparador de palavras de 4 *bits* em que a saída seja 1 somente quando A ≠ B. Use 4 portas XOR e obtenha um diagrama com um total de 5 portas, como o da **Figura 2**. A porta de saída será uma:
   a) NAND de 4 entradas.
   b) NOR de 4 entradas.
   c) AND de 4 entradas.
   d) OR de 4 entradas.

3. Implemente um comparador de palavras de 4 *bits* em que a saída seja 1 somente quando A = B. Use apenas portas XOR de 2 entradas. O diagrama terá no mínimo:
   a) 6 portas XOR.
   b) 7 portas XOR.
   c) 8 portas XOR.
   d) NDA

4. No comparador de palavras de 2 *bits* do item **2.2** da parte experimental, se usássemos apenas portas AND de duas entradas, portas OR de duas entradas e NOT, teríamos um circuito com (suponha que as entradas possuam seus complementos disponíveis):
   a) 7 portas AND, 4 portas OR e 4 NOT.
   b) 9 portas AND, 3 portas OR e 2 NOT.
   c) 10 portas AND, 4 portas OR e 2 NOT.
   d) 11 portas AND, 4 portas OR e 2 NOT.

5. Utilizando-se somente portas XOR, podemos implementar portas:
   a) NOT.
   b) OR e NOT.
   c) AND e NOT.
   d) NDA