



## 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:

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

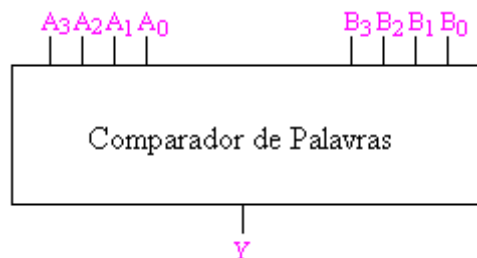
O **comprimento** de uma palavra binária é o número de *bits* que compõem a palavra. Por exemplo, a palavra binária 01101 possui 5 *bits*.

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.

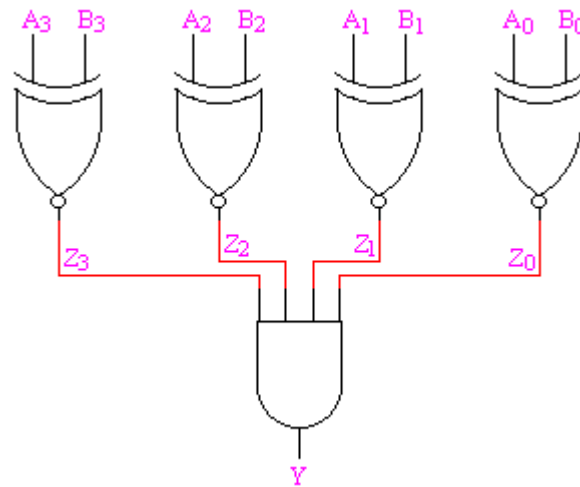
O diagrama de blocos de um comparador de palavras de 4 *bits* é mostrado na **Figura 1**.



**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  $A = B$ .

### 1.3. SOFTWARE DE SIMULAÇÃO E SÍNTESE QUARTUS-II

Nesta disciplina utilizaremos o software Quartus-II, da Altera, para realizar simulações e síntese dos circuitos digitais em FPGA.

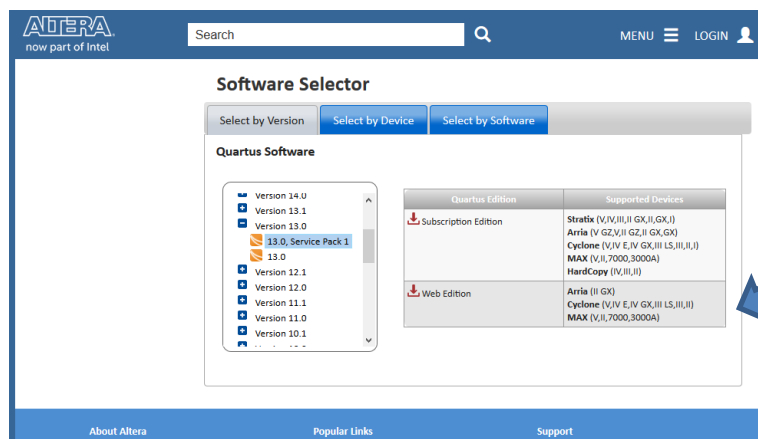
O Quartus-II é a ferramenta de desenvolvimento desenvolvido pela empresa Altera para seus produtos FPGA. Esta ferramenta permite a síntese de circuitos descritos em diagramas esquemáticos ou nas Linguagens de Descrição de Hardware Verilog, System Verilog ou VHDL, para fins de implementação no dispositivo FPGA ou para simulação.

No LINF, use a versão 13.0 do Quartus-II. (Está instalada também a versão 9.1!)

Instale o Quartus-II v13.0 (não use nenhuma versão mais nova) na sua máquina de casa ou no seu notebook. Faça o download da versão Web Edition a partir do site da Altera :

<https://dl.altera.com/13.0sp1/?edition=web&lang=en>

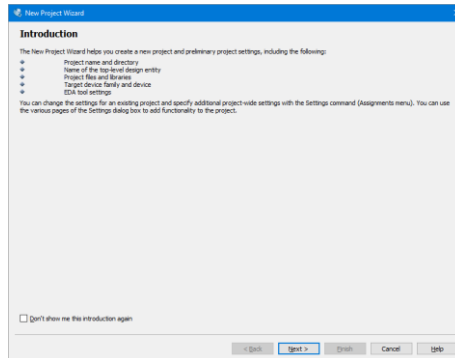
**Não instale a versão de avaliação/subscription que funciona só por 30 dias!**



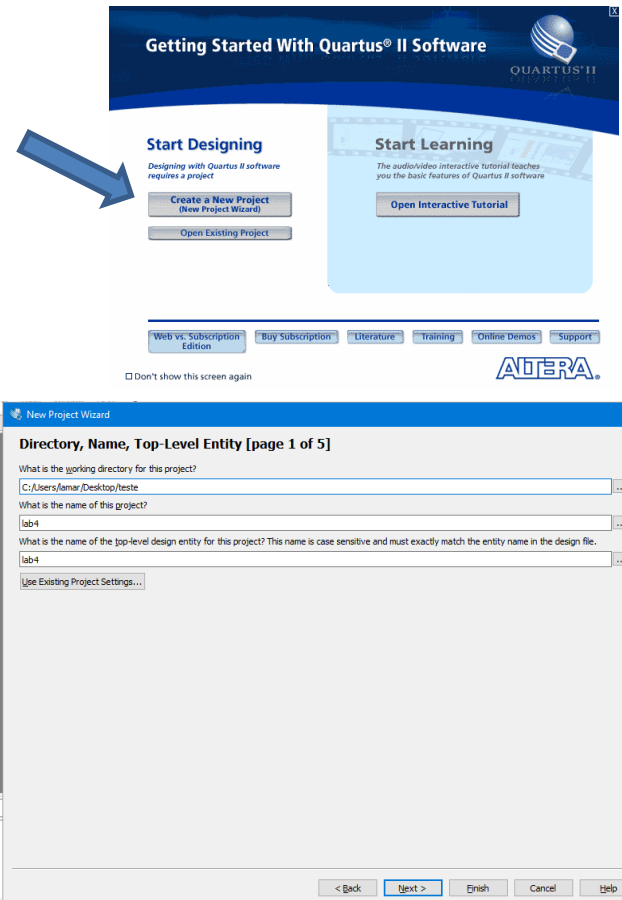
## 1) Iniciando um novo projeto

- Iniciar o programa Quartus-II através do Menu Iniciar do Windows, submenu Altera.
- Escolha criar um novo projeto: Create a New Project (New Project Wizard)

- Introduction: Next



- Directory, name and Top-Level: **Crie um novo diretório na Área de Trabalho** onde estará o projeto. Defina o nome do Projeto e, automaticamente, o nome do arquivo Top-Level. Next




**Importante:** Sempre use nomes que não contenham caracteres especiais !@#%\$%^&()-[]{}.;<>/?| etc.


- Add Files: Next – Não precisa neste momento adicionar nenhum arquivo.
- Family & device settings: Family Cyclone II : EP2C35F672C6 : Next
- EDA Tool Settings: Next – Usaremos as ferramentas padrões.
- Summary: Finish - Fim

## 2) Criando um novo Circuito Esquemático

- Selecione: File > New > Block Diagram / Schematic File

- Inserting: Pressione o ícone Symbol Tool (ícone ) para inserir um novo componente. Selecione a biblioteca Altera/Primitives/Logic.

- Placement: Escolha o local onde cada componente será colocado

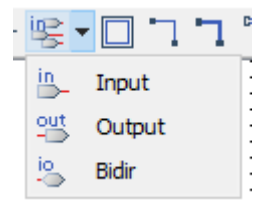
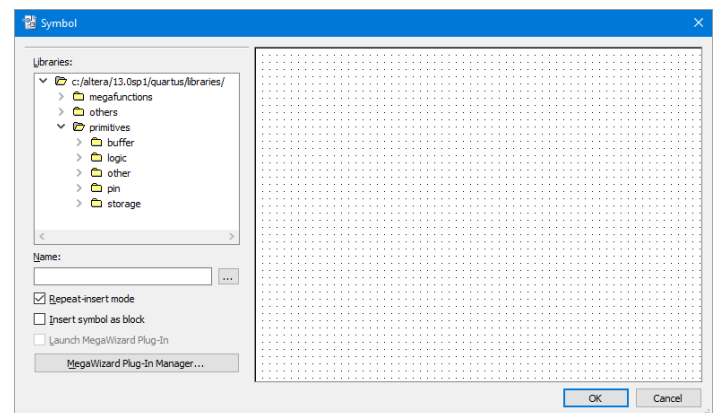
- Routing: Faça as ligações (fios) usando o ícone Orthogonal Node Tool . Cuide para que as conexões sejam feitas corretamente! Sem curtos circuitos nem maus contatos!

- Insira os Pinos de IO: Defina os pontos de entrada IN e saída OUT do circuito pelo ícone Pin Tool

Dica: Use nomes A[0], A[1], A[2] para definir uma palavra de entrada A com 3 bits.

**NÃO** dê nomes aos pinos que iniciem com Números nem que contenham caracteres especiais (!@#%\$%^&(=)-[]{}.;<>/?| "espaço" etc.)


- Salve o esquemático em um arquivo .bdf

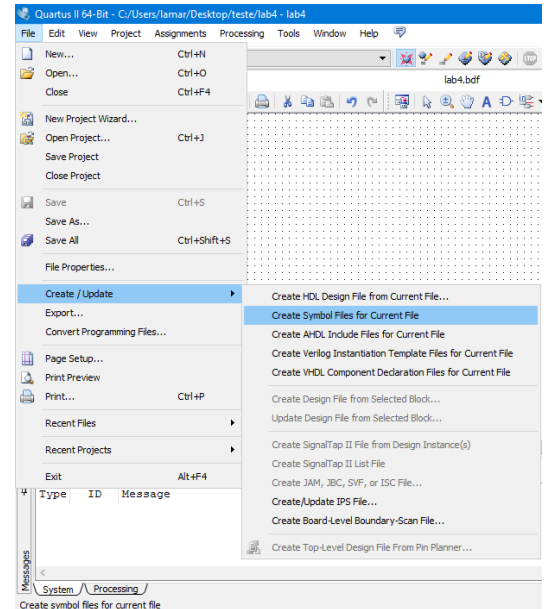


## 2.1) Criando um sub-circuito

- Crie um novo circuito File > New > Block Diagram/Schematic File
- Desenhe o sub-circuito, definindo também os nomes dos pinos de entrada e saída (não use caracteres especiais nos nomes dos pinos ><=)
- Salve o sub-circuito em formato .bdf
- Crie o símbolo: File > Create/Update > Create Symbol Files From Current file
- Salve o símbolo como arquivo .bsf


## 2.2) Usando um sub-circuito já criado em seu circuito principal (Top-Level)

- Inserir novo componente ícone Symbol Tool 
- Selecionar a biblioteca que está no diretório Project
- Lá deve estar o símbolo criado para o seu sub-circuito



## 3) Compilando o projeto

Lembre-se que o arquivo principal do projeto, que será compilado e simulado inicialmente, é definido pelo Top-Level!

- Inicie a compilação ícone Start Compilation 
- Corrija os erros encontrados e "despreze" os *Warnings*.
- Os requerimentos físicos do projeto:
  - número de elementos lógicos/registradores/memória/multiplicadores podem ser obtidos a partir do relatório da compilação Compilation Report > Flow Summary
  - tempos tpd,tsu,tco,th podem ser obtidos a partir do relatório da compilação Compilation Report > QuestTime Timing Analyzer > Multicorner Report

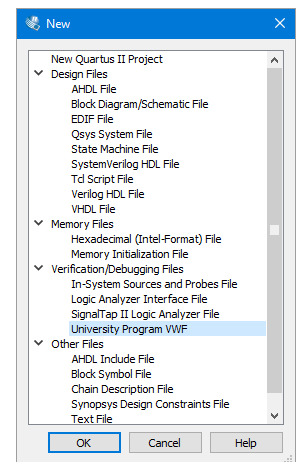
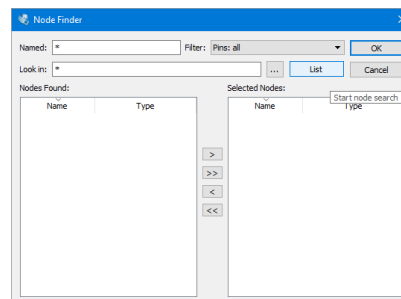
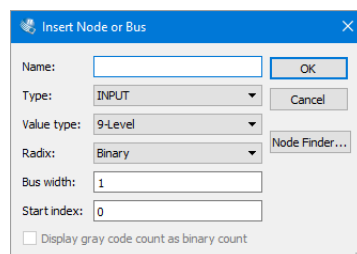
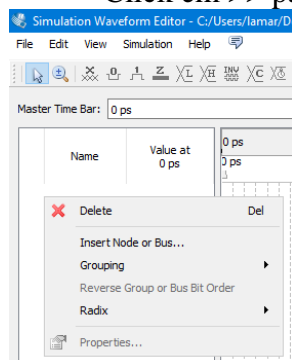
## 4) Simulando o projeto

- Crie o arquivo com a forma de onda dos sinais de entrada: File > New > University Program (VWF)
- Defina os pinos a serem considerados na simulação.

Click com botão direito > Insert Node or Bus > Node Finder > Selecione Filter:

Pins All > List

Click em >> para adicionar todos os nós encontrados.



- Crie os sinais que verificam toda a tabela verdade:

-Selecione e agrupe os sinais de entrada e click em Count Value



- Defina o período (em ns) de cada contagem (ex.: 40 ns): Edit/Set End Time/ define o tempo total da simulação

- Salve como arquivo .vwf

- Execute as simulações

-Definir o Simulador: Simulation > Options > Quartus-II Simulator

-ícone Run Functional Simulation: simulação funcional, portas lógicas ideais

-ícone Run Timing Simulation: simulação temporal, considerando os tempos de atrasos

- Tire *print screen* das formas de onda e do circuito para colocar no relatório.

## 2. PARTE EXPERIMENTAL

2.1. **(Pós-Experimento 1)** Projetar **(Pré-Projeto 1)** e simular um comparador de palavras de 3 *bits*,  $A=A_{[2]}A_{[1]}A_{[0]}$  e  $B=B_{[2]}B_{[1]}B_{[0]}$ , usando apenas portas NAND de duas entradas.

a) Complete a tabela da verdade abaixo do circuito XNOR e obtenha a sua função booleana  $Z_i$ .

Entradas		Saída
$A_i$	$B_i$	$Z_i$
0	0	
0	1	
1	0	
1	1	

- b) Modifique a função  $Z_i$  obtida no item **a** de forma a usar apenas portas NAND de 2 entradas.
- c) Desenhe, como um subcircuito, o diagrama lógico parcial da comparação do par de bits ( $A_i$ ,  $B_i$ ), realize a simulação funcional e verifique a tabela da verdade do item **a**.
- d) Desenhe o circuito total e faça a simulação funcional do diagrama lógico total, verificando a tabela verdade obtida.
- e) Realize a simulação temporal do circuito e comente os resultados obtidos justificando-os.

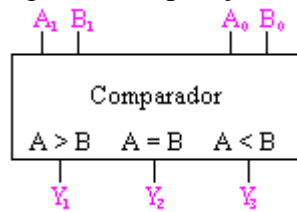
2.2. **(Pós-Experimento 2)** Projetar **(Pré-Projeto 2)** um comparador de duas palavras de 2 *bits*, com 3 saídas, tal que  $Y_1 = 1 \Leftrightarrow A > B$ ,  $Y_2 = 1 \Leftrightarrow A = B$  e  $Y_3 = 1 \Leftrightarrow A < B$ .

Pode usar as portas lógicas AND, OR, NOT e XOR.

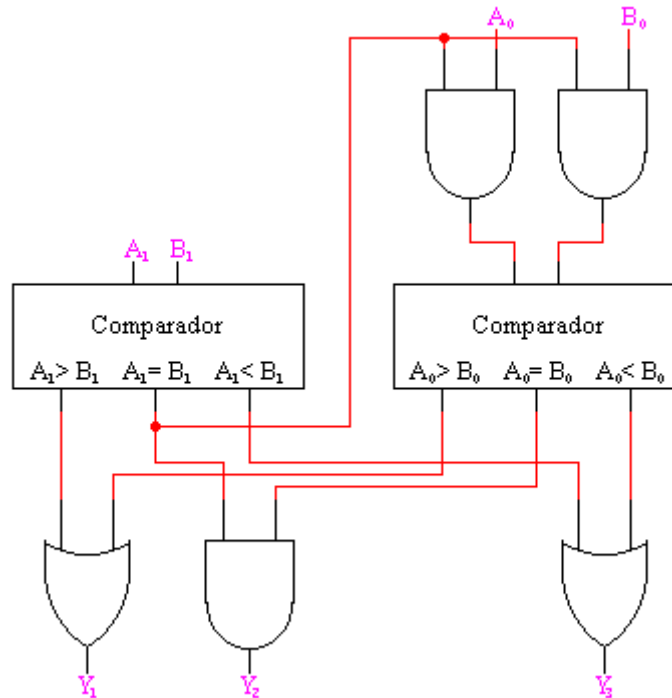
$$A = A_{[1]}A_{[0]}$$

$$B = B_{[1]}B_{[0]}$$

Divida o problema em dois comparadores de 1 *bit*. A primeira comparação é feita com os *bits* mais significativos. Se forem iguais é feita a segunda comparação conforme as figuras 3 e 4.



**Figura 3 – Comparador de 2 palavras de 2 bits com 3 saídas**



**Figura 4 –Implementação utilizando comparadores de 1 bit**

- Elabore a tabela da verdade parcial e obtenha as funções booleanas parciais do circuito Comparador de 1 bit.
- Minimize as expressões obtidas no item **a**.
- Desenhe o subcircuito e faça a simulação funcional do diagrama lógico parcial do Comparador de 1 bit, comparando com a tabela verdade do item **a**.
- Desenhe o circuito e realize a simulação funcional do diagrama lógico total do Comparador de 2 bits, escrevendo a tabela verdade obtida.
- Realize a simulação temporal do circuito do Comparador de 2 bits e comente os resultados.

### 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  $A < B$ . Foi apresentado ainda uma introdução ao software Quartus-II, usado para simulação e síntese de circuitos em FPGA.

#### 4. EQUIPAMENTOS E MATERIAL

- software Quartus II versão 13.0

#### 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  $A \neq B$ . 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