Universidade Federal de Santa Catarina
Centro Tecnológico – CTC
Departamento de Engenharia Elétrica

http://gse.ufsc.br

“EEL7020 – Sistemas Digitais”

Prof. Eduardo Augusto Bezerra

Eduardo.Bezerra@eel.ufsc.br

Florianópolis, março de 2013.

Plano de Aula

“Projeto de Sistemas Digitais com VHDL”

- Objetivos:
  - Apresentar uma visão geral de VHDL
  - Exemplo de descrição VHDL
  - Introdução ao Quartus II – ferramentas de desenvolvimento
  - Estudo de caso / exercício

VHDL - Visão Geral

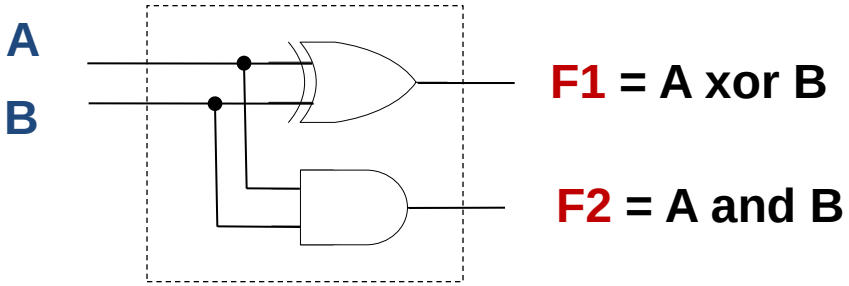
- VHDL - linguagem para descrição de hardware
- VHDL = VHSIC Hardware Description Language
- VHSIC = Very High Speed Integrated Circuits. Programa do governo dos USA do início dos anos 80.
- No final da década de 80, VHDL se tornou um padrão IEEE (Institute of Electrical and Electronic Engineers).
- Existem diversas ferramentas para simular e sintetizar (gerar hardware) circuitos descritos em VHDL.
- Outras linguagens de descrição de hardware: Verilog, SystemC, AHDL, Handel-C, System Verilog, Abel, Ruby, ...

VHDL - Visão Geral

- **O projeto de um circuito digital pode ser descrito em VHDL em diversos níveis de abstração (estrutural, comportamental).**
- **Descrições no nível de transferência entre registradores (RTL, Register Transfer Level) são bastante utilizadas.**
- **VHDL NÃO é uma linguagem de programação, e as ferramentas de síntese (não são de "compilação") não geram códigos executáveis a partir de uma descrição VHDL.**
- **Descrições em VHDL podem ser simuladas (executadas em um simulador).**
- **Descrições em VHDL podem ser utilizadas para gerar um hardware (arquivo para configuração de um FPGA, por exemplo).**
- **A geração de estímulos para simulação VHDL é realizada por intermédio de testbenches.**
- **Um testbench define os estímulos externos a serem utilizados como entrada para o circuito (definição do comportamento externo ao circuito sob teste).**
- **O testbench pode ser escrito em VHDL ou em diversas outras linguagens (ex. C, C++, ...).**

Descrição de circuito digital em VHDL

ENTITY



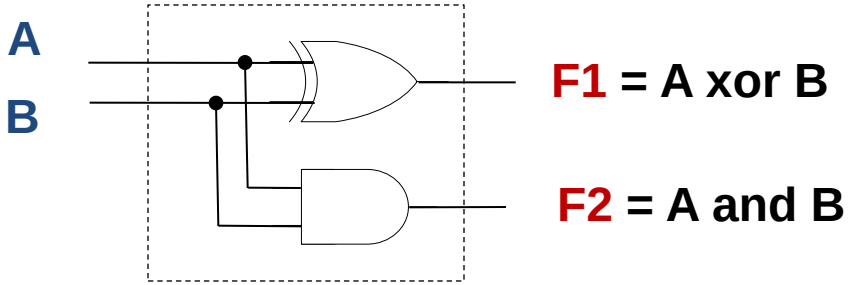
A	B	F1	F2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

```
entity halfadd is
port (A: in std_logic;
      B: in std_logic;
      F1: out std_logic;
      F2: out std_logic
      );
end halfadd;
```

ENTITY – define os "pinos" do circuito digital (sinais), ou seja, a interface entre a lógica implementada e o mundo externo.

Descrição de circuito digital em VHDL

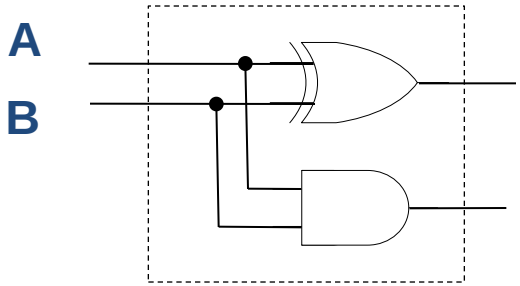
ARCHITECTURE




```
architecture circuito_logico of halfadd is
begin
   F1 <= A xor B;
   F2 <= A and B;
end circuito_logico;
```

ARCHITECTURE – define a funcionalidade do circuito digital, utilizando os "pinos" de entrada e saída listados na ENTITY em questão. Uma ENTITY pode possuir diversas implementações diferentes (diversas ARCHITECTURES).

Descrição completa do circuito em VHDL (Entity e Architecture)




```
library IEEE;
use IEEE.Std_Logic_1164.all;
```

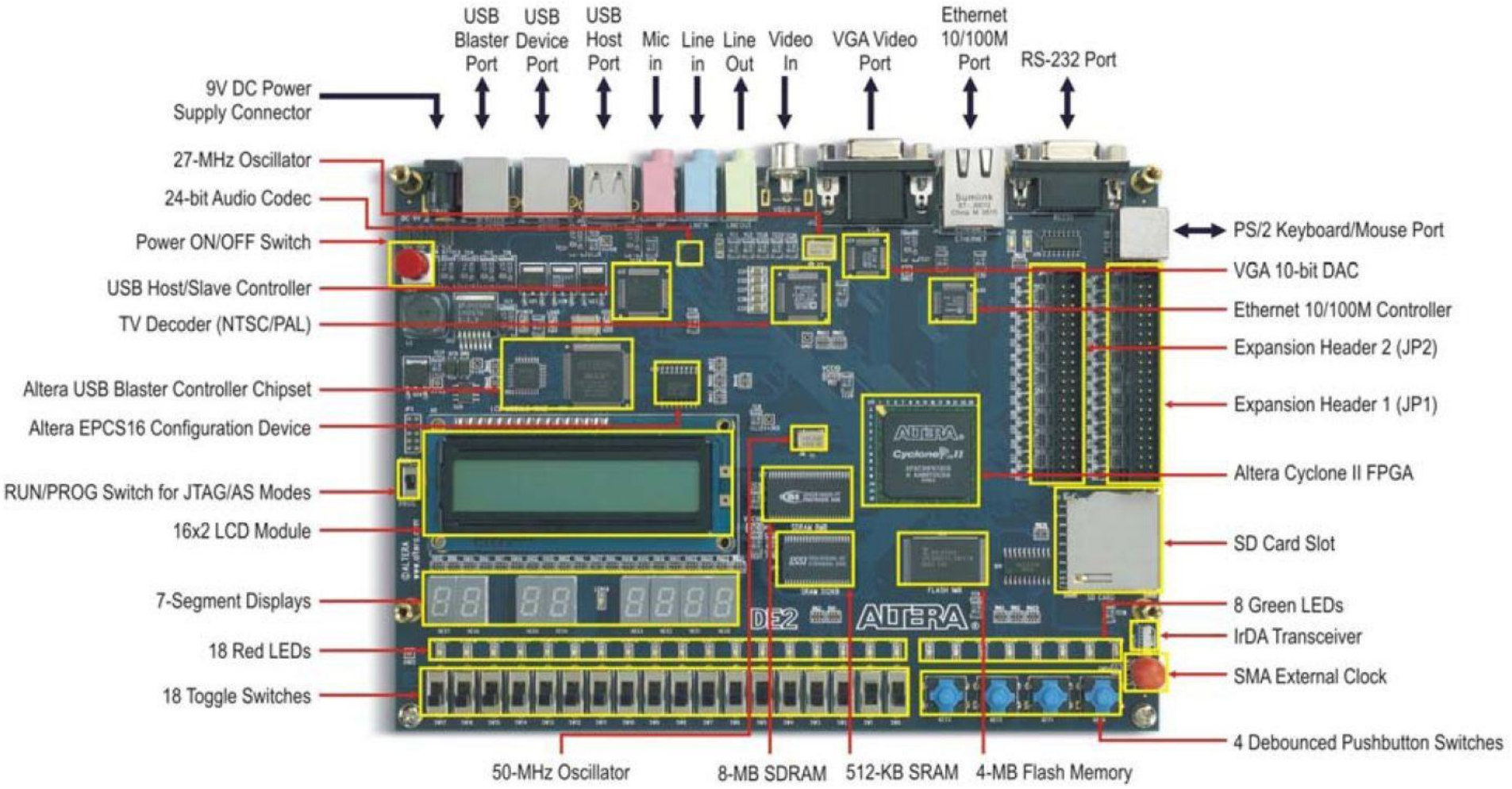
```
entity halfadd is
port (A: in std_logic;
      B: in std_logic;
      F1: out std_logic;
      F2: out std_logic
      );
end halfadd;
```

Para utilizar o tipo std_logic é necessário incluir um pacote da biblioteca IEEE.

```
architecture circuito_logico of halfadd is
begin
  F1 <= A xor B;
  F2 <= A and B;
end circuito_logico;
```

Plataforma de prototipação FPGA Altera - DE2

Kit DE2 da Altera



Interface com o usuário (entrada e saída)

- **Placa DE2 possui 18 LEDs vermelhos denominados LEDR$_{17-0}$ e 18 chaves denominadas SW$_{17-0}$**
- **As conexões entre esses componentes e os pinos do FPGA da placa estão definidas no arquivo *DE2_pin_assignments.csv***
- **São utilizados "vetores" para facilitar o acesso aos LEDs e chaves da placa**
- **Exemplo: SW[0] é o elemento 0 do vetor SW, e está conectado ao pino PIN_N25 do FPGA**
- **No código em VHDL, usar sempre os nomes definidos no arquivo *DE2_pin_assignments.csv* (ver Pinos.csv no site da disciplina)**

Interface com o usuário (entrada e saída)

- **Código VHDL para "leitura" das chaves e "escrita" nos LEDs**

```
library ieee;
use ieee.std_logic_1164.all;
entity part1 is
      port ( SW     : in std_logic_vector(17 downto 0);
             LEDR : out std_logic_vector(17 downto 0)
           );
end part1;
architecture behavior of part1 is
begin
      LEDR(7 downto 0) <= SW(15 downto 8);
      LEDR(15 downto 8) <= "01010101";
      LEDR(17) <= (SW(17) AND SW(0)) OR (SW(16) AND '1');
end behavior;
```

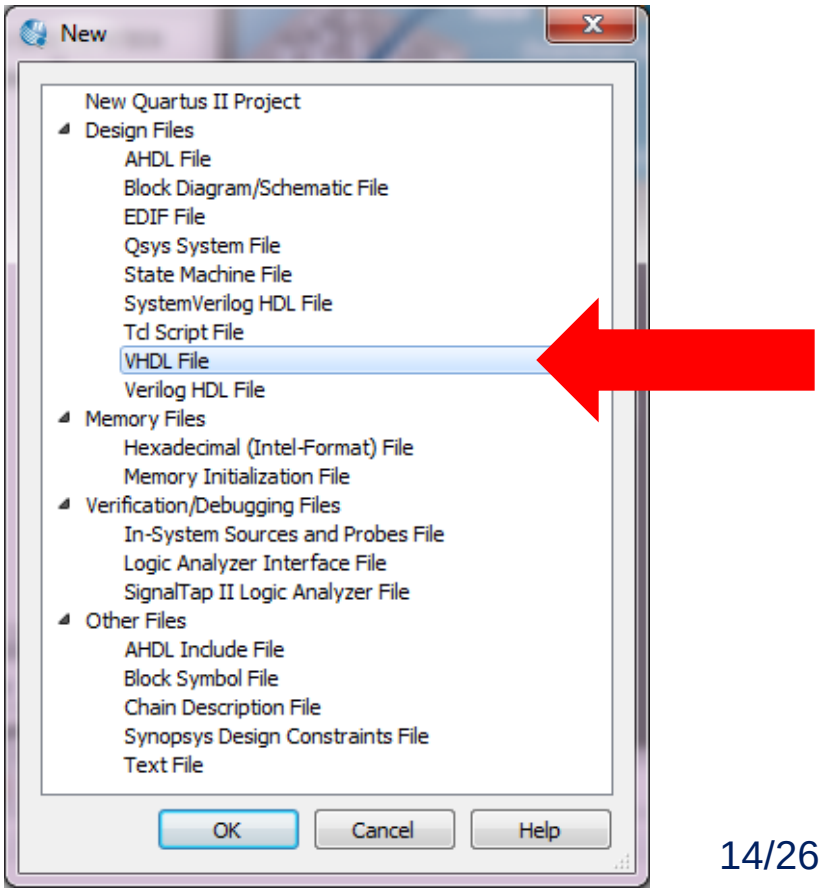
Tarefa a ser realizada na aula prática

Tarefa a ser realizada na aula prática

1. **Utilizando a ferramenta Quartus II da Altera, criar um projeto VHDL que implemente o circuito apresentado no slide 7 (*and* e *xor*).**
2. **Realizar a simulação do circuito (VHDL) por intermédio de diagramas de formas de onda, e obter a tabela verdade.**
3. **Visando fixar o conhecimento do fluxo de ferramentas de projeto, seguir o tutorial descrito no livro texto, e detalhado na última aula.**
4. **Utilizando as dicas do slide 11, alterar o projeto de forma a realizar a entrada dos dados A e B a partir das chaves SW(0) e SW(1), e a apresentação dos resultados F1 e F2 no LEDR(0) e LEDR(1).**
5. **Testar o circuito no kit DE2, usando as chaves SW(0) e SW(1) para entrar com os operandos, e observar os resultados nos LEDs.**

Resumo do tutorial: *Etapa 1 - Design Entry*

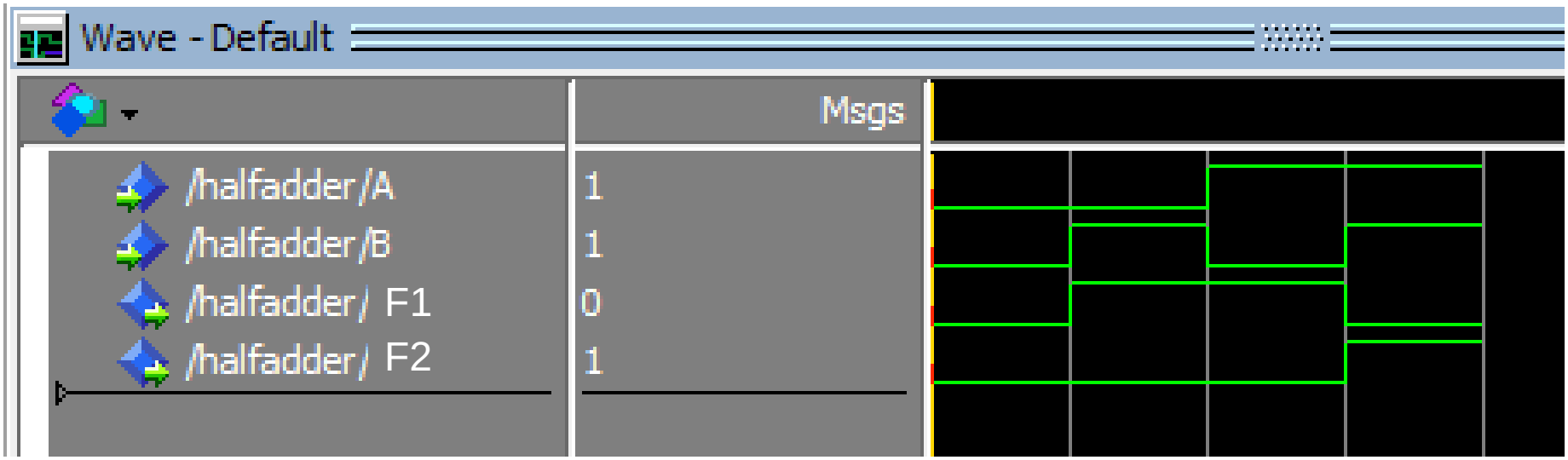
1. [Quartus II] File -> New Project Wizard
2. No “project wizard”, seguir exatamente os passos do tutorial da última aula.
3. File -> New -> VHDL File (Essa é a principal diferença!).
4. Copiar o fonte VHDL do slide 7 para o novo arquivo, e salvar.



Resumo do tutorial: *Etapa 2 - Simulação*

5. **[ModelSim] Simulação Funcional – Teste do circuito**
   -> não considera informação de temporização.
   - *Seguir o tutorial de simulação da última aula.*

6. **Resultado esperado da simulação:**



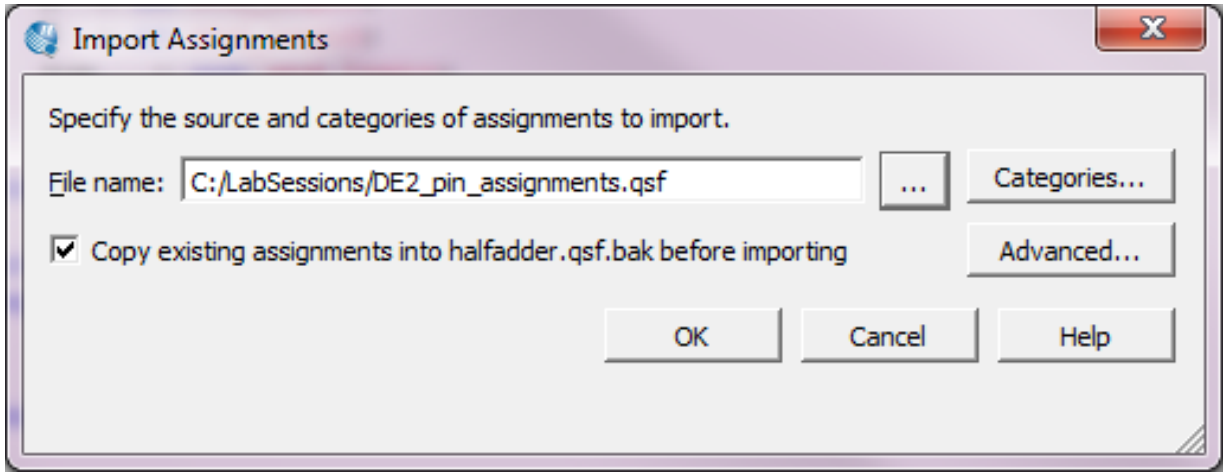
Resumo do tutorial: *Etapa 3 – prototipação FPGA*

7. Adaptar o fonte para os nomes de sinais da DE2:

```
1   library IEEE;
2   use IEEE.Std_Logic_1164.all;
3
4   entity halfadder is
5   port (                                                  -- A -> SW(0)
6       SW  : in  std_logic_vector(17 downto 0);            -- B -> SW(1)
7       LEDR: out std_logic_vector(17 downto 0)             -- sum   -> LEDR(0)
8       );                                                  -- carry -> LEDR(1)
9   end halfadder;
10
11  architecture ha_stru of halfadder is
12  begin
13    LEDR(0) <= SW(0) xor SW(1);       -- sum   <= A xor B
14    LEDR(1) <= SW(0) and SW(1);       -- carry <= A and B
15  end ha_stru;
```

Resumo do tutorial: *Etapa 3 – prototipação FPGA*

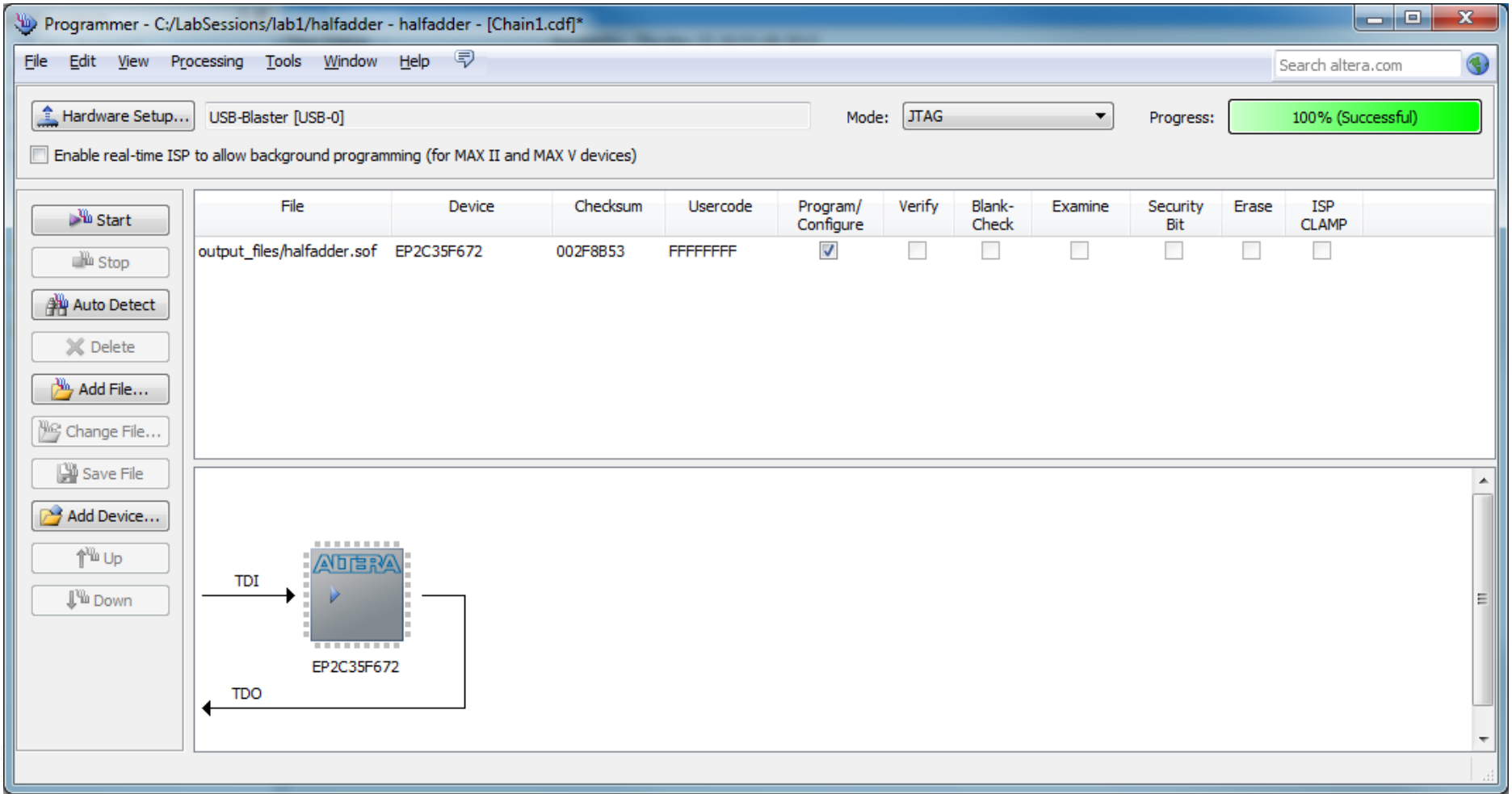
8. **Assignments -> Import Assigments (procurar no site e usar o arquivo *DE2_pin_assignments.qsf* ou *Pinos.qsf*)**



9. **Com isso, os pinos do FPGA foram associados aos sinais da entity do VHDL**
10. **Compilar o VHDL (síntese)**
11. **ATENÇÃO!!! Verificar se o nome da entity é o mesmo nome do projeto, para evitar erros na síntese.**
12. **A compilação resulta em cerca de 400 warnings devido aos pinos não conectados do arquivo *.qsf***

Resumo do tutorial: *Etapa 3 – prototipação FPGA*

13. Programação – FPGA é carregado com circuito, configurando fisicamente elementos de processamento e roteamento.
Tools – Programmer. Hardware Setup – USB-Blaster. Start!



Para ir além: uso do LCD

Escrita no LCD da placa DE2 da Altera

```
library ieee;
use ieee.std_logic_1164.all;
entity LCD is
port (
    LCD_DATA:   out std_logic_vector(7 downto 0);
    LCD_RW:     out std_logic;
    LCD_EN:     out std_logic;
    LCD_RS:     out std_logic;
    LCD_ON:     out std_logic;
    LCD_BLON:   out std_logic;
    SW :        in std_logic_vector(17 downto 0)
);
end LCD;

architecture LCD_WR of LCD is
begin
        LCD_ON      <= SW(17);
        LCD_BLON    <= SW(16);
        LCD_DATA    <= SW(7 downto 0);
        LCD_RS      <= SW(8);
        LCD_EN      <= SW(9);
        LCD_RW      <= SW(10);
end LCD_WR;
```

Passos para inicializar (CONFIGURAR) o LCD

PASSO 1: envia comando 1 (38H).

Esse comando liga o LCD, liga o cursor, e faz o cursor piscar.

Chave (SW)	Valor (posição da chave)	Efeito
17	1	LCD_ON
16	1	LCD_BLON
7 .. 0	0011 1000	Comando
8	0	LCD_RS (0 = controle)
9	$0 \rightarrow 1 \rightarrow 0$	LCD_EN
10	0	LCD_RW

- ligar LCD
- ativar cursor e piscar

Passos para inicializar (CONFIGURAR) o LCD

PASSO 2: envia comando 2 (0FH).

Esse comando liga o LCD, liga o cursor, e faz o cursor piscar.

Chave (SW)	Valor (posição da chave)	Efeito
17	1	LCD_ON
16	1	LCD_BLON
7 .. 0	0000 1111	Comando
8	0	LCD_RS (0 = controle)
9	$0 \rightarrow 1 \rightarrow 0$	LCD_EN
10	0	LCD_RW

- ligar LCD
- ativar cursor e piscar

Passos para inicializar (CONFIGURAR) o LCD

PASSO 3: envia comando 3 (06H).

Esse comando liga o LCD, liga o cursor, e faz o cursor piscar.

Chave (SW)	Valor (posição da chave)	Efeito
17	1	LCD_ON
16	1	LCD_BLON
7 .. 0	0000 0110	Comando
8	0	LCD_RS (0 = controle)
9	$0 \rightarrow 1 \rightarrow 0$	LCD_EN
10	0	LCD_RW

- ligar LCD
- ativar cursor e piscar

Procedimento de ESCRITA de caracteres no LCD

PASSO 4: envio de dados para o LCD
O LCD aceita caracteres da tabela ASCII (ver http://asciitable.com). No exemplo a seguir está sendo escrito o caracter A, ou seja, 41H, ou 01000001 em binário.

Chave (SW)	Valor (posição da chave)	Efeito
17	1	LCD_ON
16	1	LCD_BLON
7 .. 0	0100 0001	Dado 'A'
8	1	LCD_RS (1 = dados)
9	$0 \rightarrow 1 \rightarrow 0$	LCD_EN
10	0	LCD_RW

Procedimento para limpar (apagar) o LCD

PASSO 5: comando para limpar (apagar) o LCD

Chave (SW)	Valor (posição da chave)	Efeito
17	1	LCD_ON
16	1	LCD_BLON
7 .. 0	0000 0001	Comando
8	0	LCD_RS (0 = controle)
9	$0 \rightarrow 1 \rightarrow 0$	LCD_EN
10	0	LCD_RW

Escrita no LCD da placa DE2 da Altera

Tutorial

http://www.feng.pucrs.br/~jbenfica/curso/tutorial_lcd.pdf

http://www.lisha.ufsc.br/~bezerra/disciplinas/Microprocessadores/tools/LCD/LCD_APLICATIVO.html