# Laboratório de Sistemas Digitais

## Trabalho Prático nº 1

## Introdução às ferramentas e kit de desenvolvimento

***Objetivos***

- Familiarização com as ferramentas de projeto e com o kit de desenvolvimento com FPGA que vai ser usado nas aulas práticas.
- Modelação em VHDL, simulação, implementação em FPGA e teste de componentes elementares.

***Sumário***

Este trabalho prático está dividido em 4 partes. Na parte I pretende-se introduzir os aspetos básicos das ferramentas e do kit de desenvolvimento que vai usar nas aulas práticas de Laboratório de Sistemas Digitais (LSDig). Isto é feito através da construção gradual de uma porta lógica NOR a partir de uma OR e de um inversor. São abordadas as diversas fases de projeto, desde a modelação até ao teste, passando pela simulação e implementação. Na parte II introduzem-se os operadores lógicos do VHDL. Na parte III é apresentado um exemplo, baseado num comparador de igualdade, para ilustrar algumas das vantagens da linguagem de descrição de hardware VHDL, ao nível da abstração e produtividade. Finalmente na parte IV é enunciado um problema para resolução de forma autónoma pelos alunos.

***Advertências muito importantes:***

- **A placa de desenvolvimento usada nas aulas práticas de Laboratório de Sistemas Digitais possui uma FPGA e diversos componentes que se podem danificar devido a descargas de eletricidade estática, pelo que deve ser manuseada com cuidado. Em particular, não deve tocar com qualquer parte do corpo ou objetos (incluindo vestuário) nos seus contatos elétricos e conetores.**
- **No final da aula, desligue o kit e arrume-o adequadamente juntamente com os cabos e alimentador na respetiva caixa.**

***Resumo do Fluxo de Projeto para Sistemas Baseados em FPGA***

A Figura 1 ilustra o fluxo de projeto para sistemas baseados em FPGA. Os diversos passos são resumidos de seguida.

A etapa de *design entry* consiste na modelação, codificação ou introdução da funcionalidade pretendida, podendo para tal o projetista usar linguagens de descrição de hardware, diagramas esquemáticos, diagramas de transição de estado ou outros métodos. No caso de LSDig vai ser utilizada a linguagem de descrição de hardware VHDL, pelo que pode ser utilizado qualquer editor de texto para este efeito, embora por conveniência seja usado o integrado no ambiente de desenvolvimento (*Integrated Development Environment* – IDE) que vamos adotar ("Altera Quartus II").

Depois de modelado o sistema, o passo seguinte é a sua síntese (lógica), isto é, a compilação do modelo de forma a criar uma *netlist* (i.e. um conjunto de portas lógicas, flip-flops,

multiplexadores, outros componentes e respetivas interligações) que implementa a funcionalidade pretendida. Este passo é realizado por ferramentas de software normalmente desenvolvidas pelo fabricante da FPGA usada.

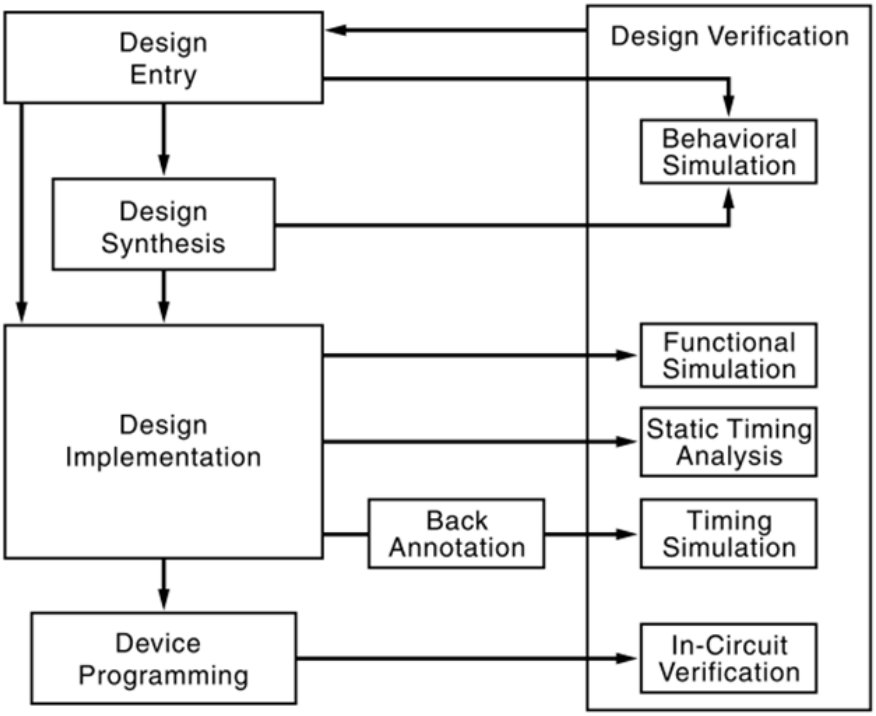


Figura 1 – Fluxo de projeto para sistemas digitais baseados em FPGA.

Seguidamente, a *netlist* vai ser implementada (compilada) para uma determinada família de FPGAs por uma ferramenta de software (desenvolvida pelo fabricante da FPGA) que realiza os seguintes passos:

- Mapeamento da *netlist* em primitivas da FPGA (tabelas de verdade, portas lógicas, multiplexadores, registos, etc.);
- Posicionamento das primitivas em localizações específicas da FPGA;
- Encaminhamento das interconexões (estabelecimento das ligações) entre as primitivas da FPGA.

O resultado da implementação é um ficheiro de programação da FPGA que deve ser usado para a sua configuração através de software e um cabo de programação adequado.

A validação do sistema pode ser realizada em diversas etapas do fluxo de projeto, por simulação ou verificação em hardware real. São também normalmente disponibilizadas pelo fabricante diversas ferramentas de análise temporal, energética, recursos lógicos utilizados, etc. A generalidade destes passos vai ser ilustrada ao longo deste trabalho prático.

## *Parte I*

1. Abra a aplicação "Altera Quartus II" e crie um novo projeto (menu *"File->New Project Wizard"*) de acordo com os passos seguintes (Figuras 2 a 7).

**Nota importante:** devido a limitações de velocidade da rede no acesso ao diretório pessoal em arca.ua.pt, nos PCs das salas de aula recomenda-se a utilização de uma *pendrive* para guardar os seus projetos e ficheiros. Caso isso não seja possível, guarde e aceda ao seu trabalho a partir de um diretório da drive Z:. Em qualquer dos casos não utilize espaços nem caracteres especiais (e.g. acentos) nas *paths* dos projetos e ficheiros, uma vez que isso causará problemas na utilização das ferramentas (isto significa que não deve gravar os seus projetos em sub-diretórios do "Ambiente de Trabalho" ou do "Meus Documentos").

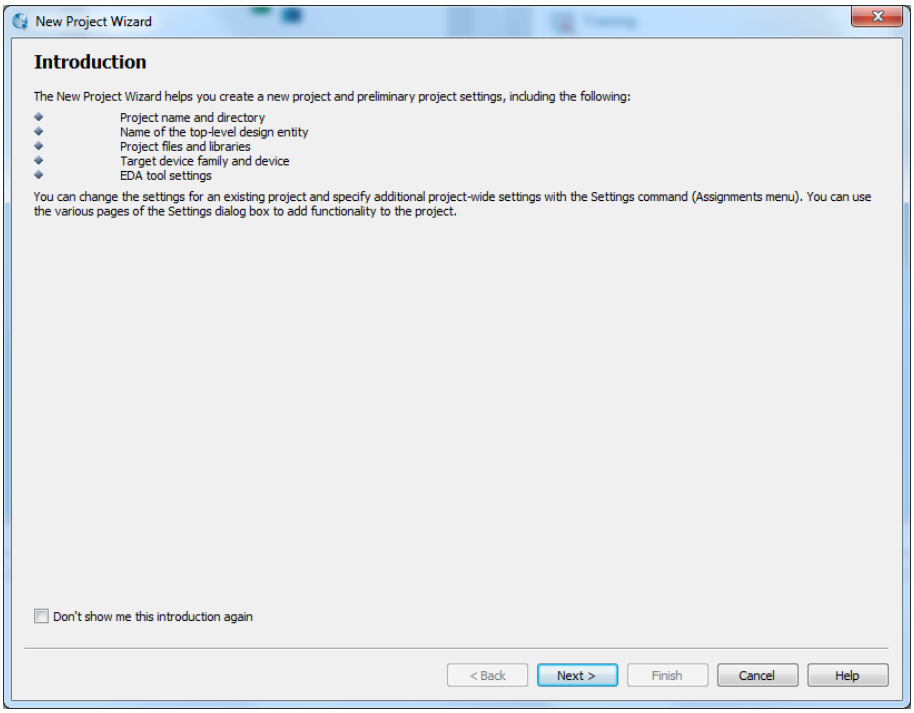


Figura 2 – Passo inicial introdutório (pode ser desativado).

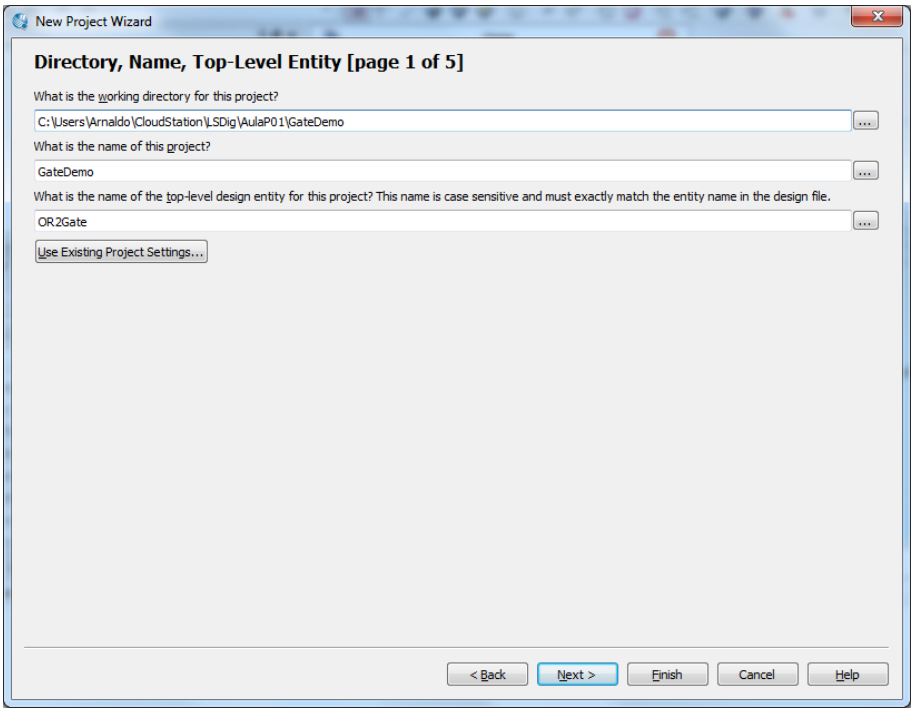


Figura 3 – Passo 1 – identificação e localização do projeto no sistema de ficheiros (adaptar de acordo com o diretório usado, o qual não deve conter espaços nem caracteres especiais, e.g. acentos, no caminho).

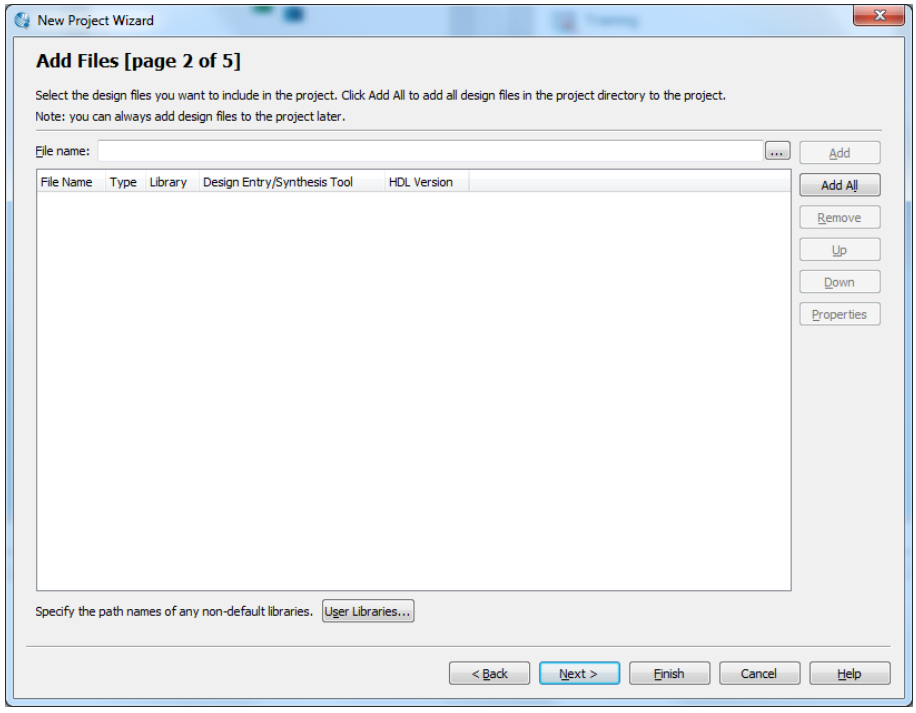


Figura 4 – Passo 2 – adição de ficheiros pré-existentes (não usado neste projeto).

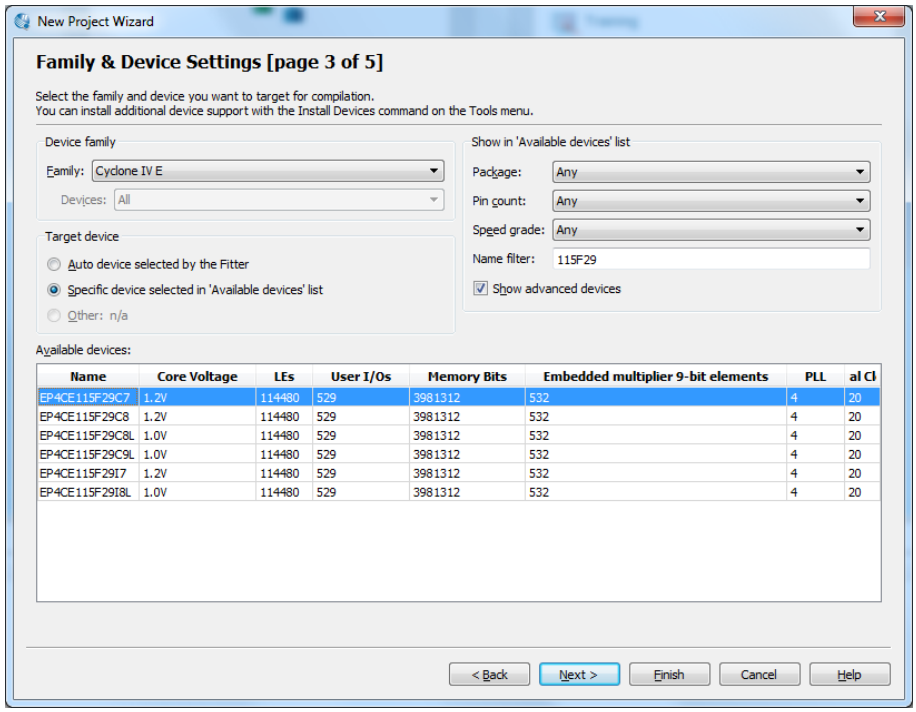


Figura 5 – Passo 3 – seleção da FPGA usada (**Altera Cyclone IV EP4CE115F29C7**) – a especificação do filtro "115F29" facilita a seleção da FPGA correta (primeira da lista).

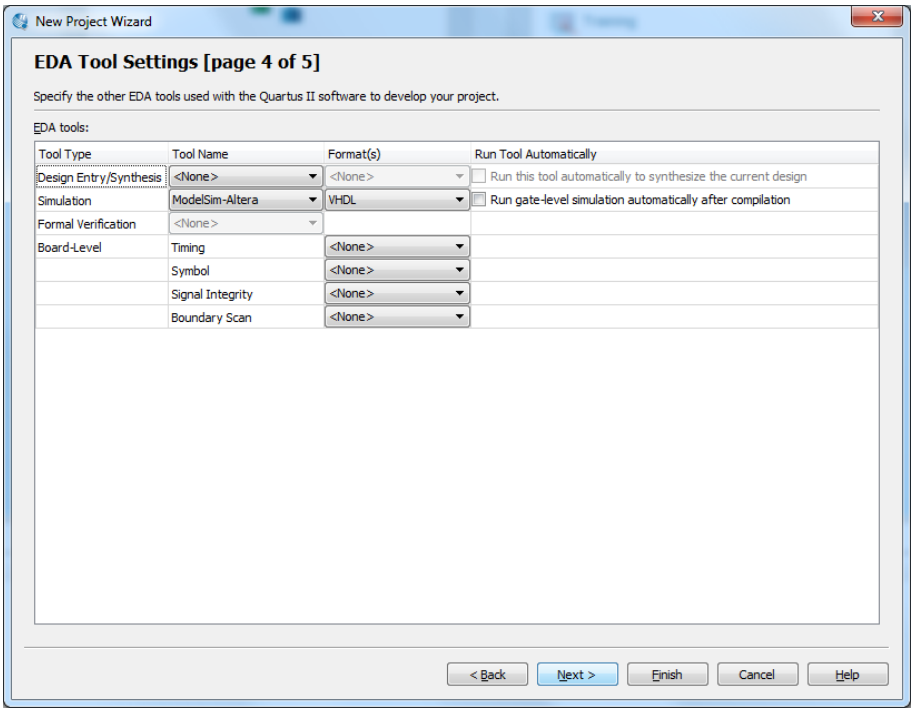


Figura 6 – Passo 4 – seleção das ferramentas e linguagens usadas no fluxo de projeto.

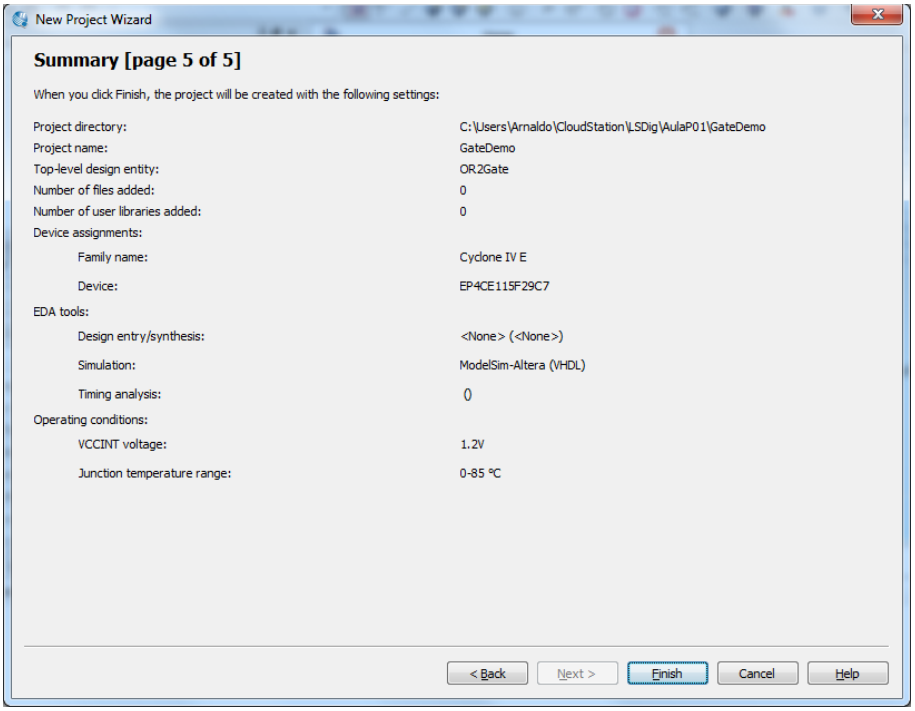


Figura 7 – Passo 5 – sumário final da criação do projeto.

2. Após premir "*Finish*" o IDE "Altera Quartus II"deve apresentar o aspeto da Figura 8.

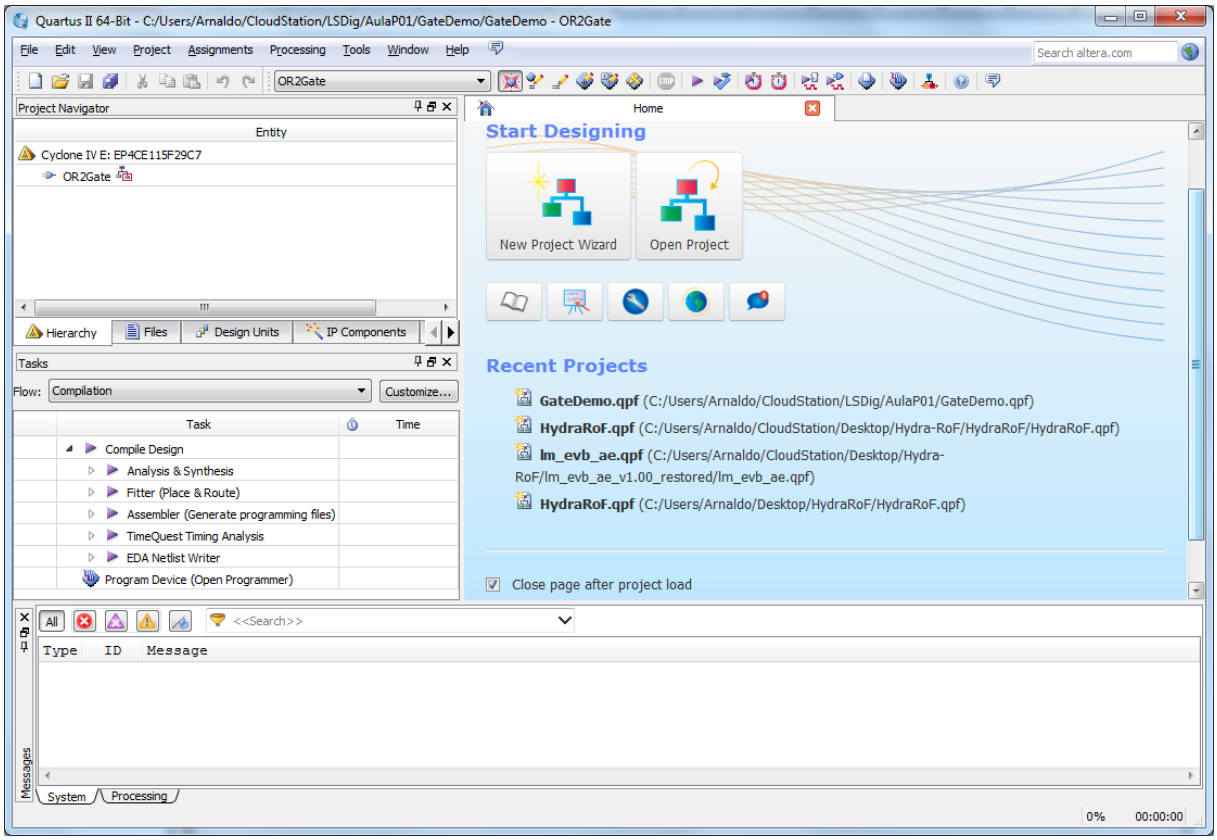


Figura 8 – Altera Quartus II IDE após a criação do projeto.

3. Crie um novo ficheiro para código fonte VHDL (menu *"File->New"*), de acordo com a Figura 9.

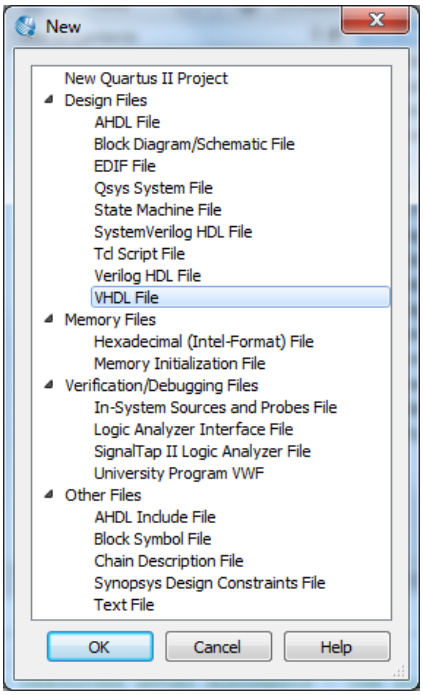


Figura 9 – Seleção do tipo de ficheiro a criar (*VHDL File*).

4. Introduza no ficheiro que acabou de criar o código VHDL correspondente a uma porta lógica OR de 2 entradas (mostrado na Figura 10).

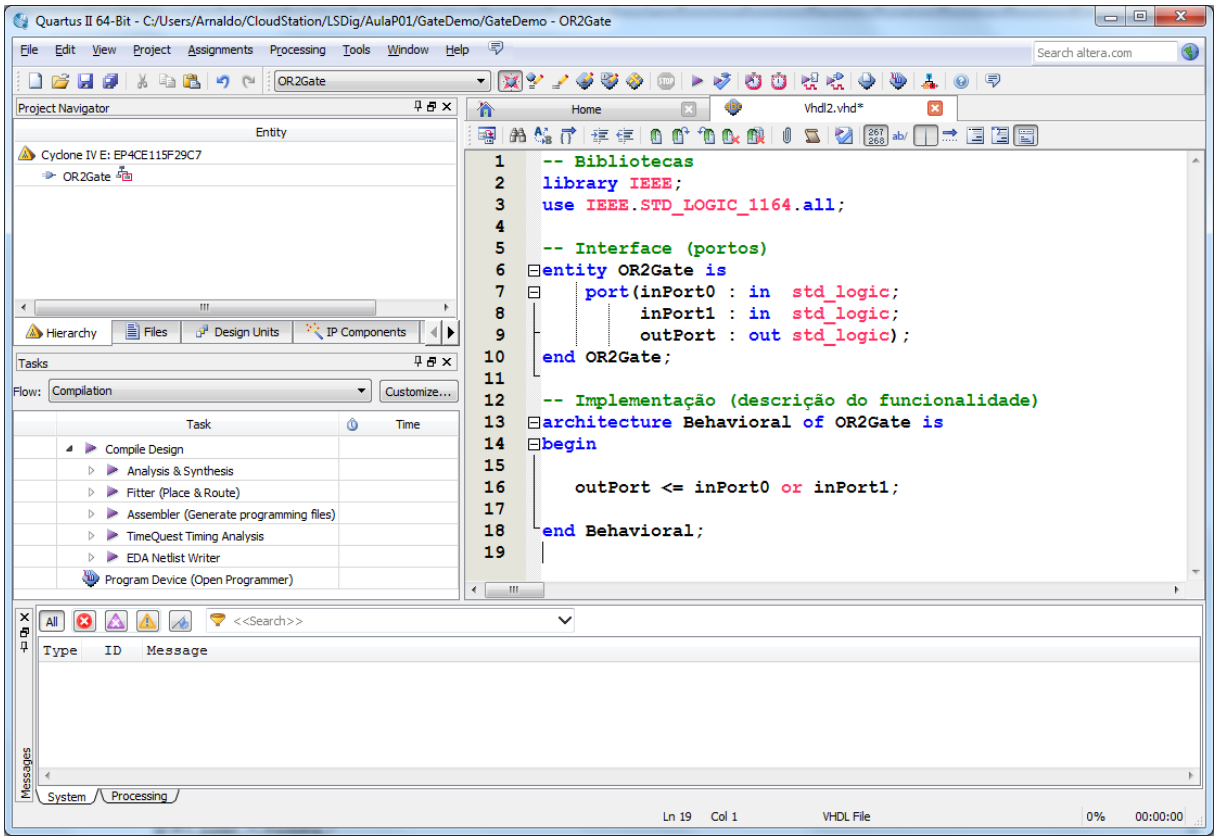


Figura 10 – Código fonte da porta lógica OR de 2 entradas (“OR2Gate.vhd”).

5. Grave o ficheiro, cujo nome deverá ser “OR2Gate.vhd” (Figura 11).

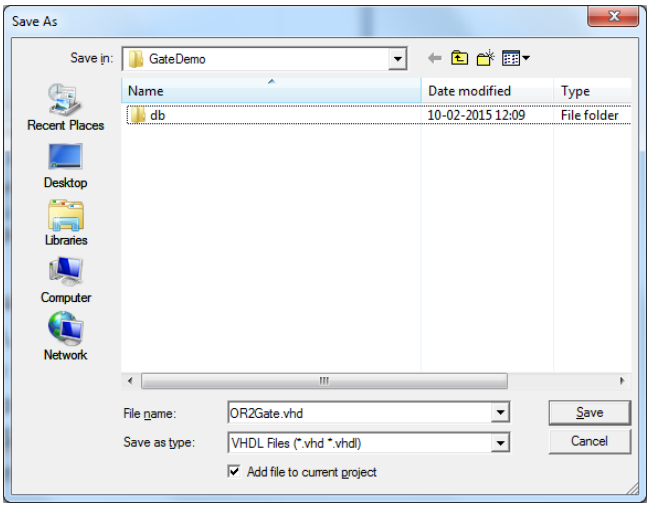


Figura 11 – Caixa de diálogo para gravação do ficheiro “OR2Gate.vhd”.

6. Seguidamente vai ser validado por simulação o comportamento da porta lógica modelada. No entanto, antes de efetuar a simulação, execute a opção “*Analysis & Synthesis*” para que a correção sintática e estrutura do projeto sejam analisadas. Após a execução da “*Analysis & Synthesis*” o IDE deve apresentar o aspeto da Figura 12.
**Nota:** Certifique-se que a opção “*ModelSim-Altera*” disponível no menu “*Tools->Options*” – “*EDA Tool Option*” possui o valor “C:\altera\13.1\modelsim_ase\win32aloem” (a primeira vez que executa a aplicação, a *string* configurada é “…ae…” em vez de “…ase…”).

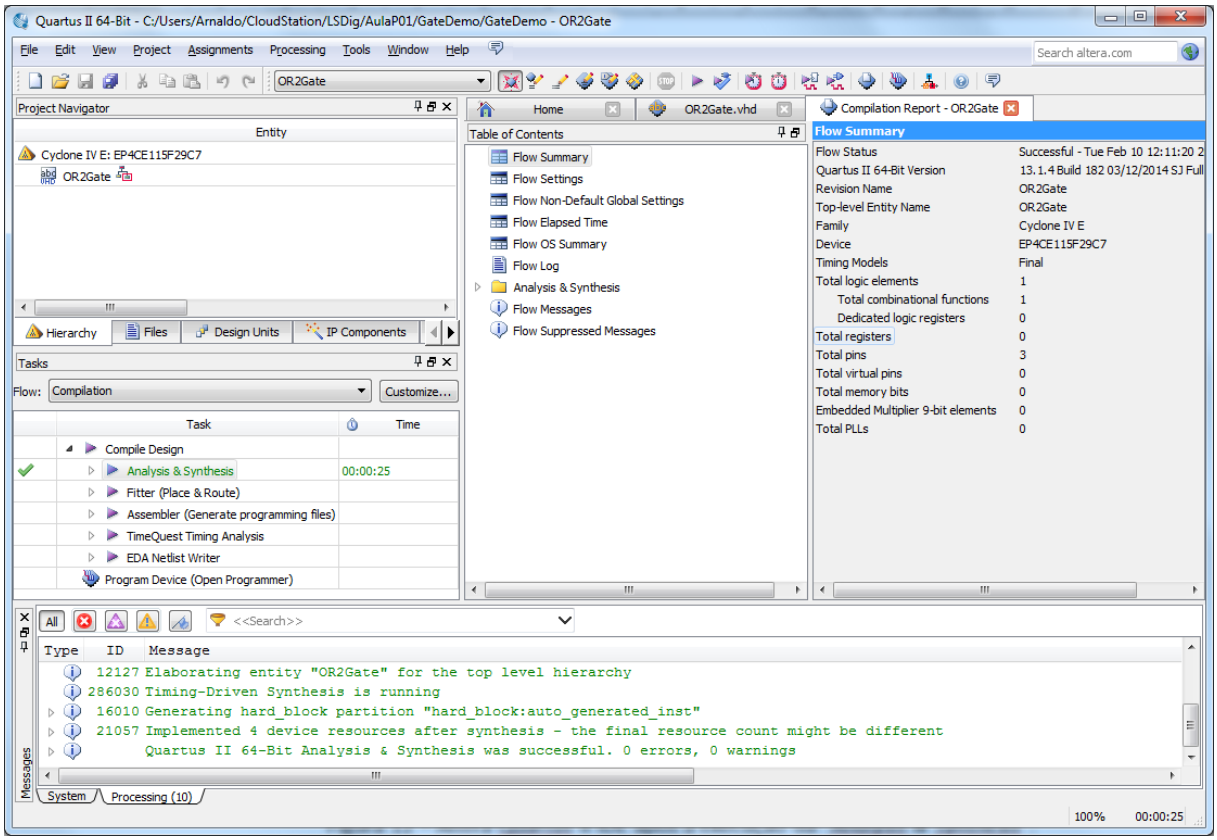


Figura 12 – Altera Quartus II IDE após a execução da "*Analysis & Synthesis*".

7. Simule o comportamento da porta lógica que acabou de descrever, criando para tal um ficheiro VWF (menu "*File->New*") de acordo com os passos seguintes (Figuras 13 a 17).

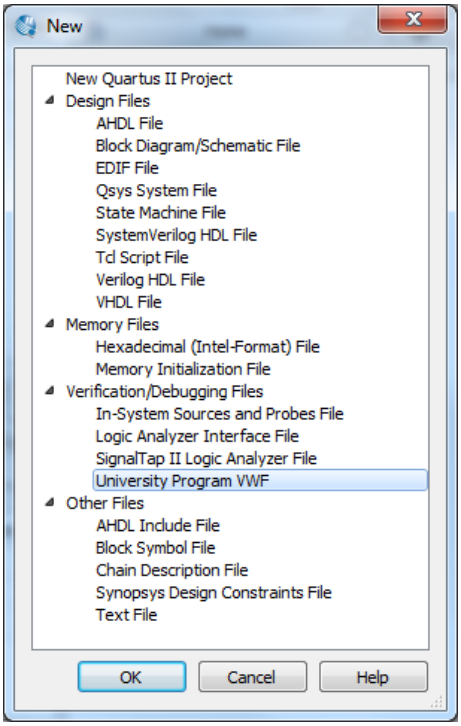


Figura 13 – Seleção do tipo de ficheiro a criar (*University Program VWF*).

8. Após premir "OK" a janela seguinte deverá abrir, onde deverão ser indicados os sinais usados na simulação (Figura 14).

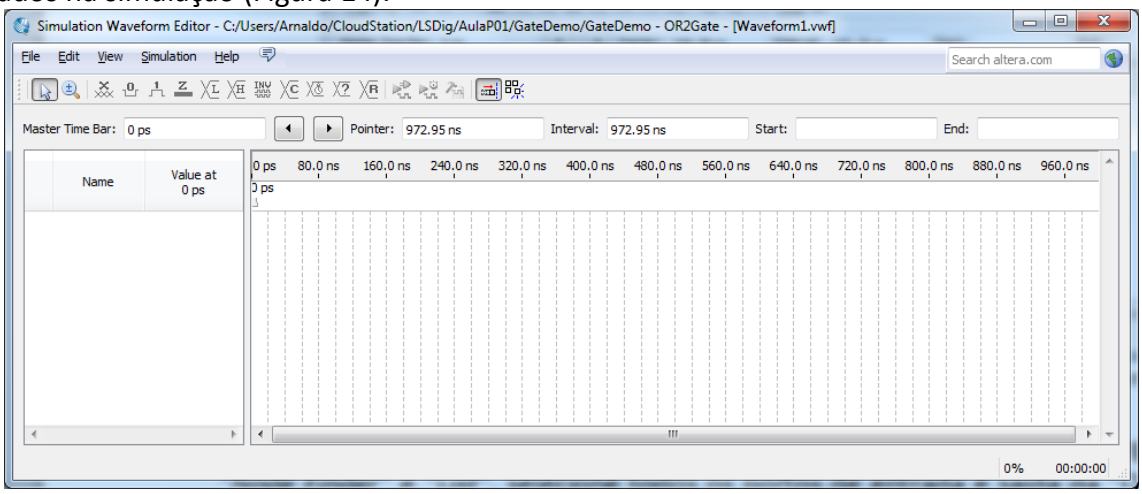


Figura 14 – Janela do simulador antes da especificação das formas de onda de entrada.

9. Através do menu *"Edit->Insert->Insert Node or Bus"*, premindo de seguida os botões *"Node Finder"* e *"List"*, selecione todos os portos de entrada e saída da "OR2Gate" (Figura 15).

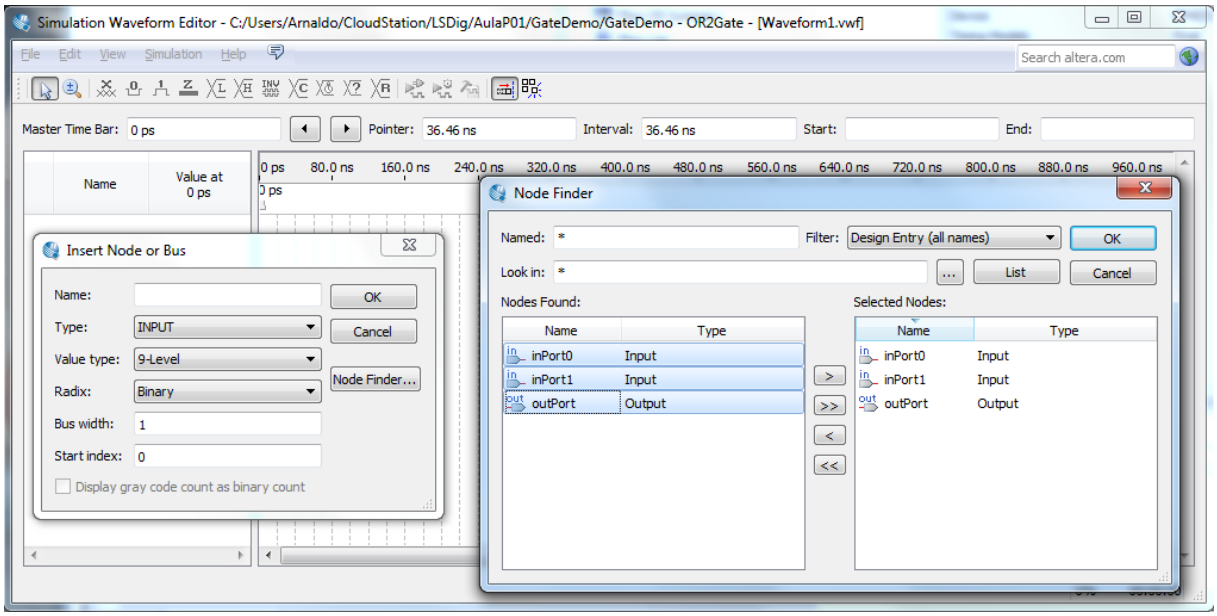


Figura 15 – Especificação das formas de onda de entrada na janela do simulador.

10. Após premir "OK" poderá especificar os valores pretendidos ao longo do tempo para as entradas da "OR2Gate". Utilize para tal o rato, selecionando os troços do diagrama temporal do sinal pretendido com o valor lógico que deseja que ele assuma (Figura 16).

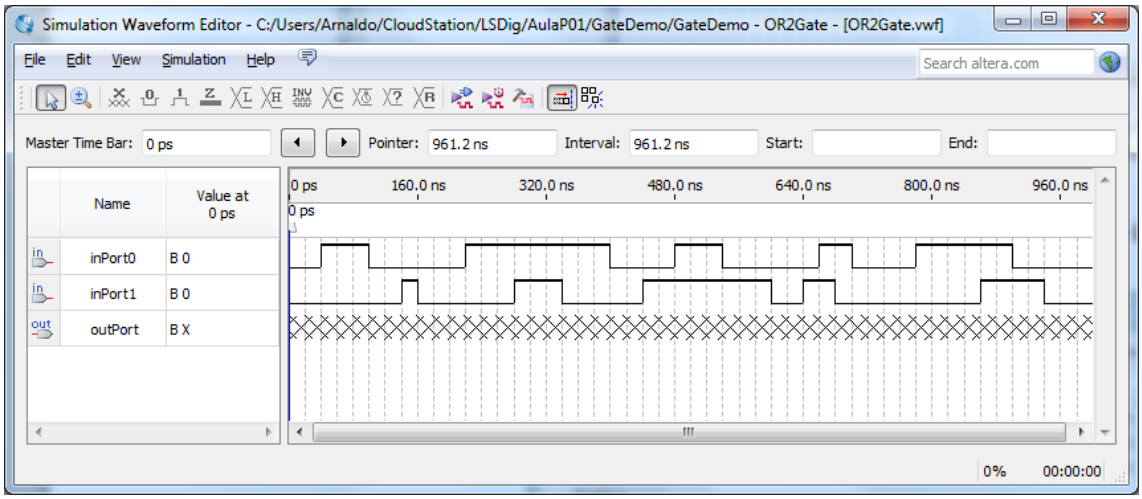


Figura 16 – Janela do simulador após a especificação das formas de onda de entrada.

11. Após os especificar os valores (vetores) de simulação, grave o ficheiro (Figura 17).

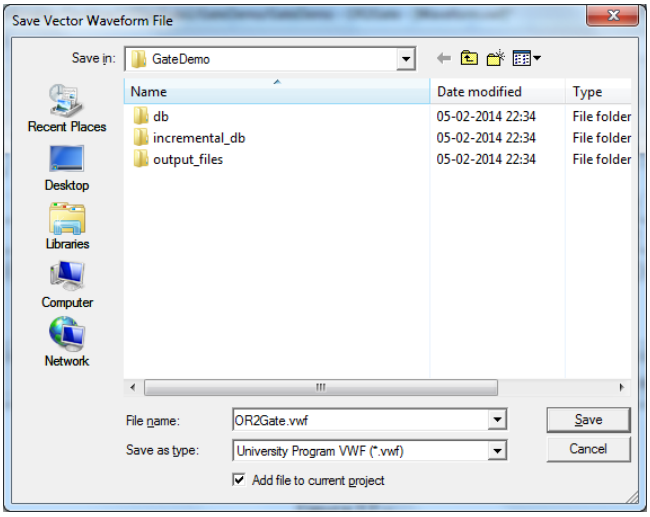


Figura 17 – Caixa de diálogo para gravação do ficheiro "OR2Gate.vwf".

12. Execute a simulação através do menu "*Simulation->Run Functional Simulation*". Após a simulação deve obter o valor da saída da porta lógica OR correspondente às entradas que especificou (Figura 18).

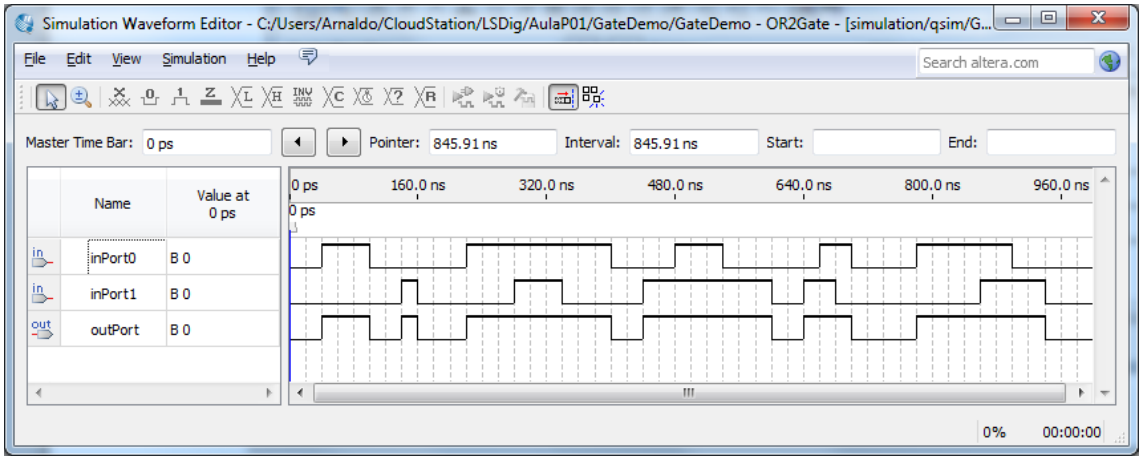


Figura 18 – Janela do simulador após a simulação.

13. Crie um novo ficheiro VHDL, chamado "GateDemo.vhd", que irá servir para instanciar a porta lógica que acabou de descrever nos pontos anteriores e associá-la a pinos concretos da FPGA do kit de desenvolvimento que vai usar para a testar.

14. Introduza o código VHDL mostrado na Figura 19 no ficheiro que acabou de criar, para instanciar a porta lógica e ligá-la a pinos adequados da FPGA (neste caso as entradas vão ser ligadas a interruptores e a saída ligada a um LED).

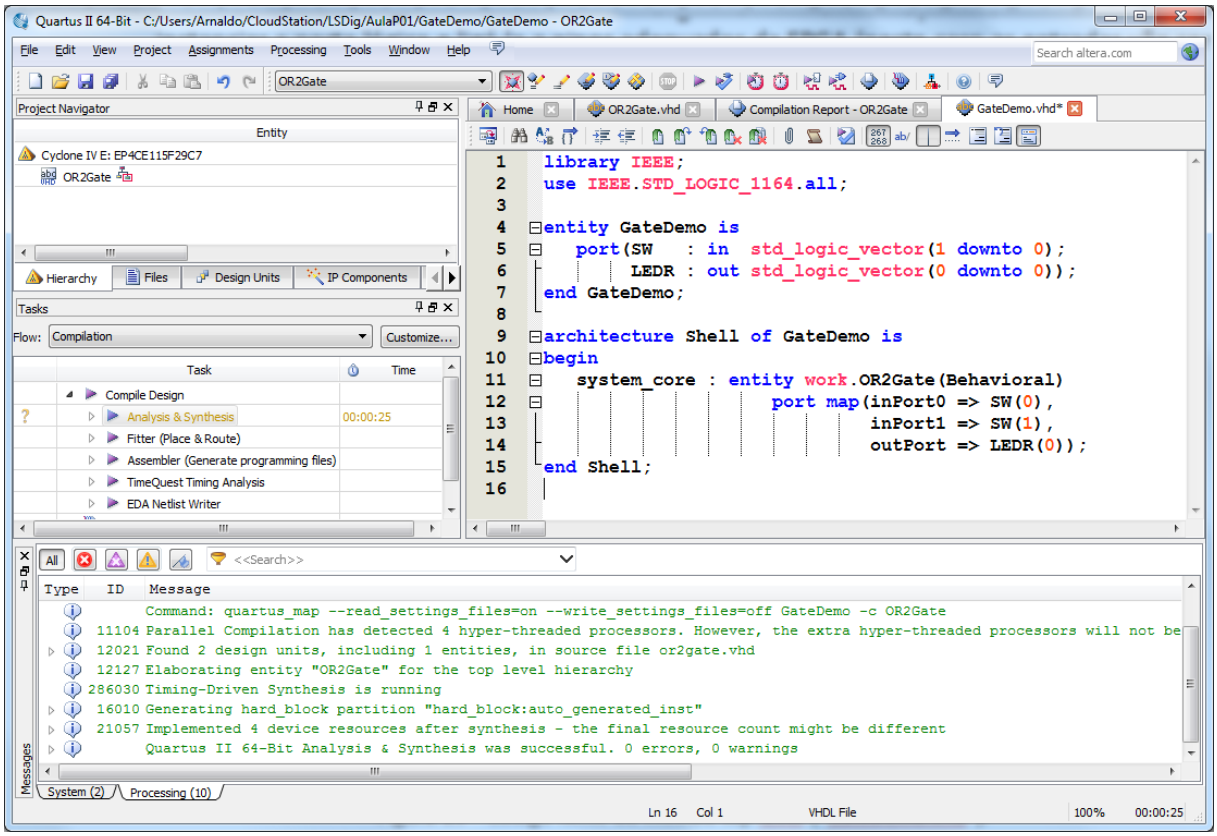

Figura 19 – Código fonte do módulo top-level ("GateDemo.vhd").

15. Após editar, grave o ficheiro (Figura 20).

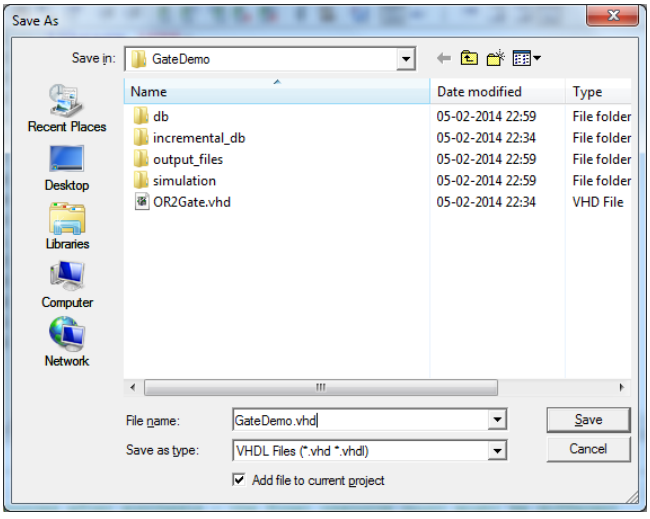

Figura 20 – Caixa de diálogo para gravação do ficheiro "GateDemo.vhd".

16. Altere o ficheiro “Top Level” do projeto (ficheiro “principal” que inclui todos os outros e portanto está no nível hierárquico mais elevado do projeto) de acordo com a Figura 21. O “Top Level” deixa de ser o “OR2Gate.vhd” e passa a ser o “GateDemo.vhd”.

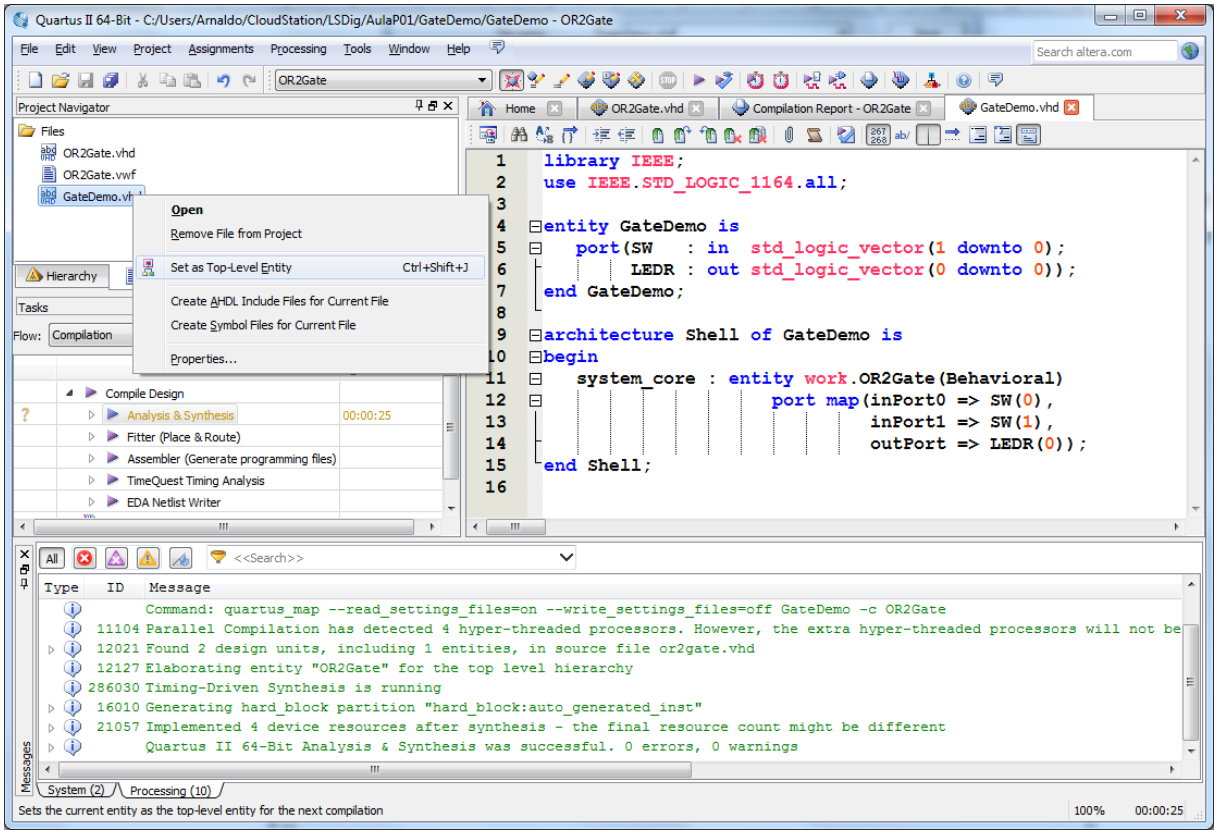


Figura 21 – Alteração do módulo *top level* para o “GateDemo.vhd”.

17. Importe as definições de pinos da FPGA da placa de desenvolvimento (localização física dos pinos aos quais estão ligados à FPGA os vários dispositivos da placa – e.g. LEDs, interruptores, displays, etc.). Para tal use o menu “*Assignments->Import Assignments*” (Figuras 22 e 23).

O ficheiro que contém todas as definições de pinos da FPGA da placa DE2-115 é o “DE2_115.qsf”. Este ficheiro está disponível para *download* no site da UC (em elearning.ua.pt) e não deve ser alterado.

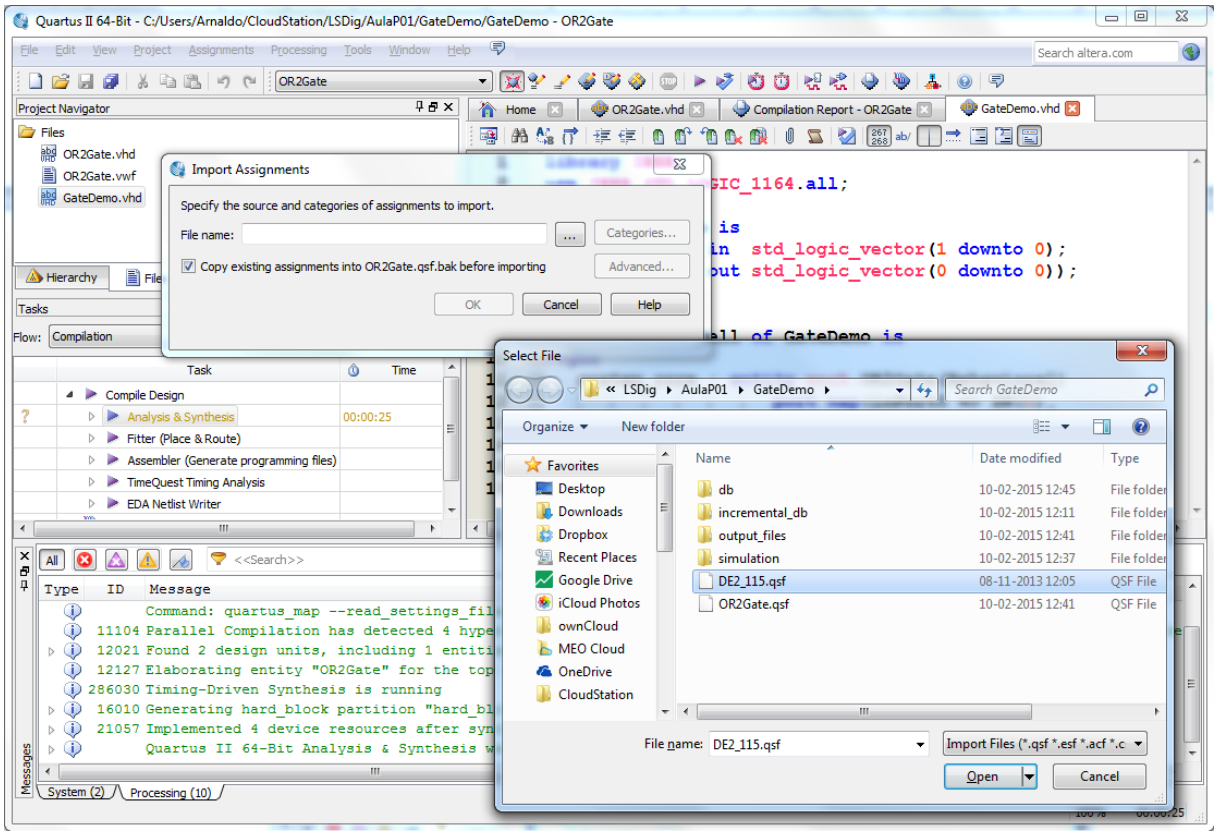


Figura 22 – Seleção do ficheiro "DE2_115.qsf" com as definições dos pinos da FPGA ligados aos dispositivos do kit.

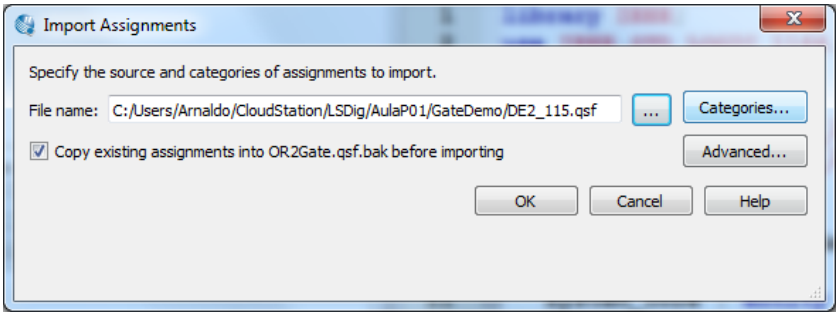


Figura 23 – Importação do ficheiro "DE2_115.qsf" com as definições dos pinos da FPGA ligados aos dispositivos do kit.

18. Efetue a síntese e implementação do projeto através do comando "*Compile Design*". No final da compilação o IDE deve apresentar o aspeto da Figura 24.

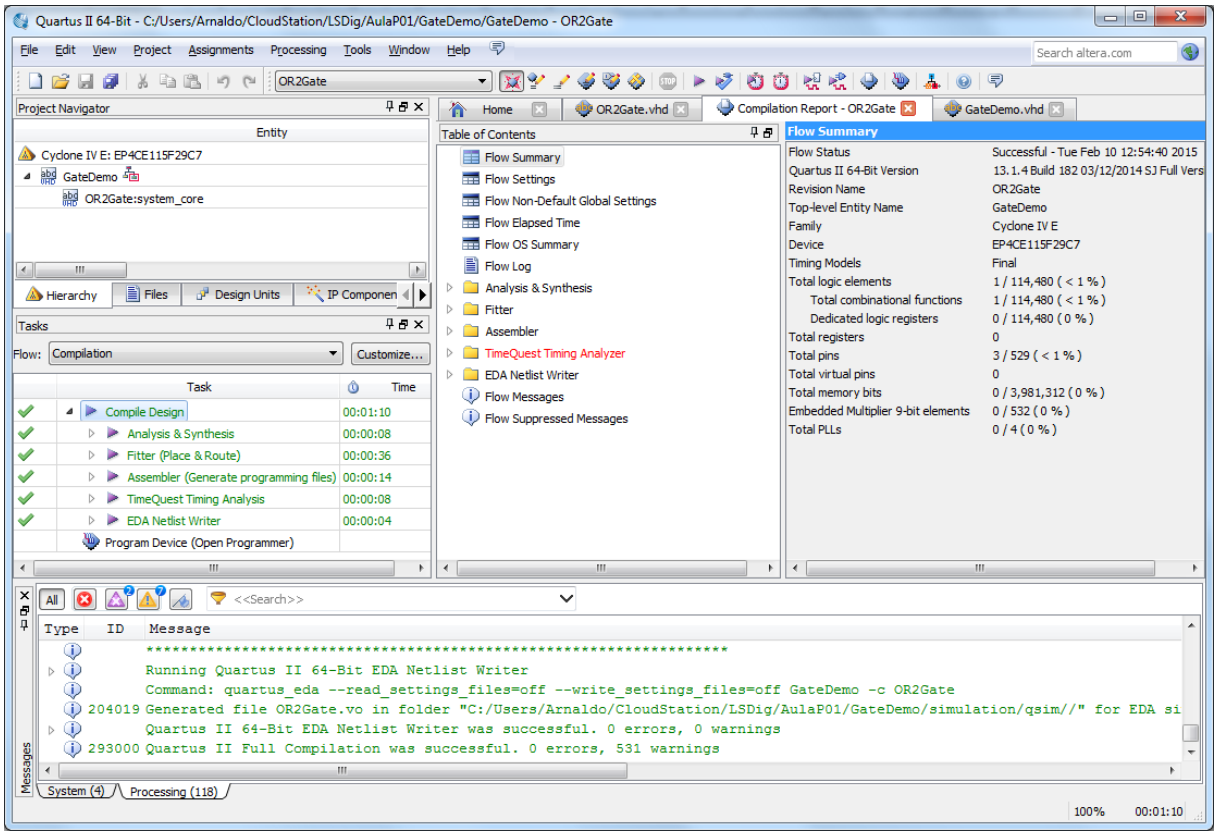


Figura 24 – Quartus II IDE após compilação (implementação) completa do projeto.

19. No final do processo de compilação, programe a FPGA através do comando "*Program Device*" que deverá abrir a janela da Figura 25.

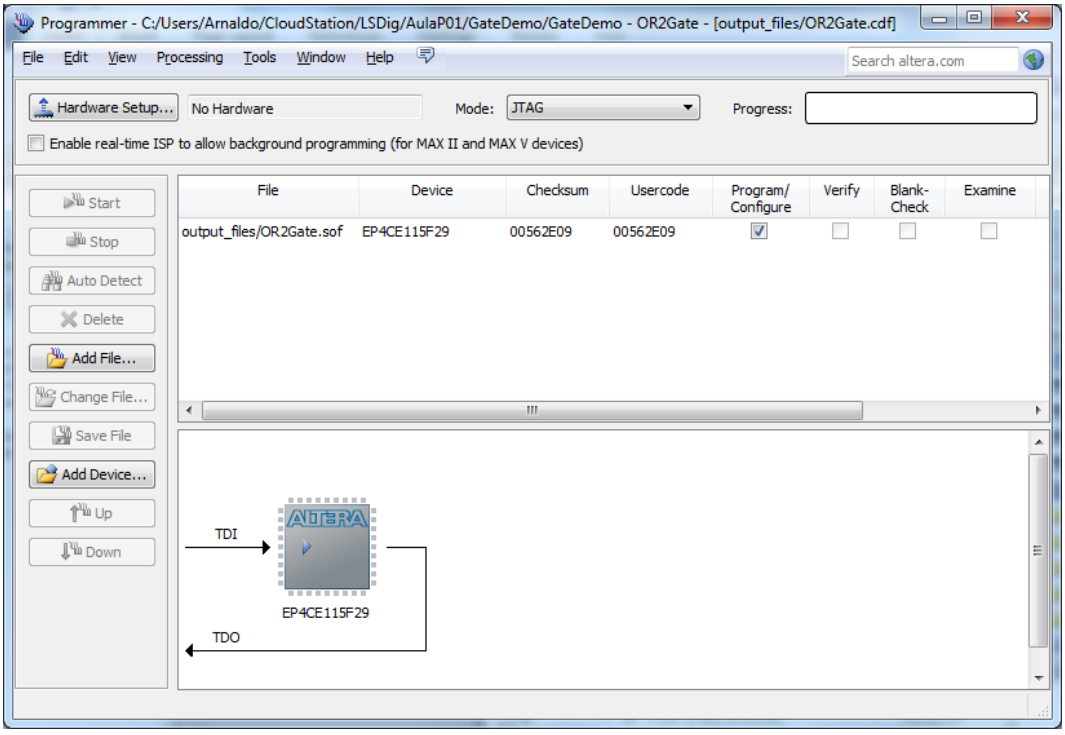


Figura 25 – Janela inicial da aplicação de programação da FPGA.

20. Configure a interface usada para programação da FPGA, premindo o botão "Hardware Setup" e selecionando a opção "USB-Blaster" (Figura 26). Caso esta opção não esteja disponível, verifique se a placa está ligada ao computador através do cabo USB.

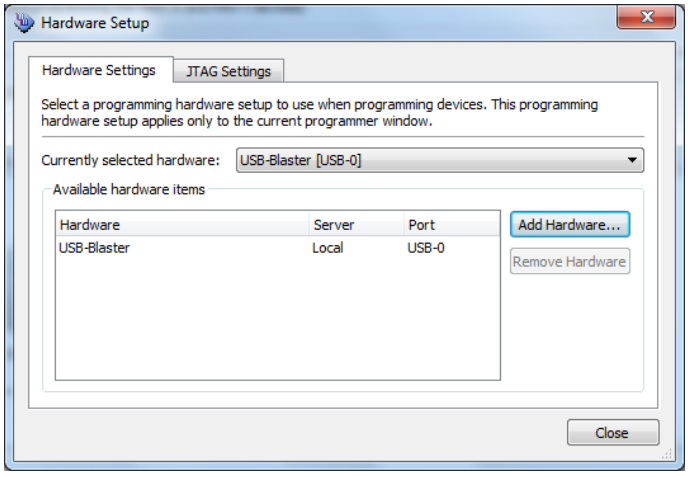


Figura 26 – Configuração da interface de programação da FPGA.

21. Para programar a FPGA prima o botão "*Start*". Quando estiver concluída (com sucesso) a programação da FPGA, a aplicação de configuração deve apresentar o aspeto da Figura 27.

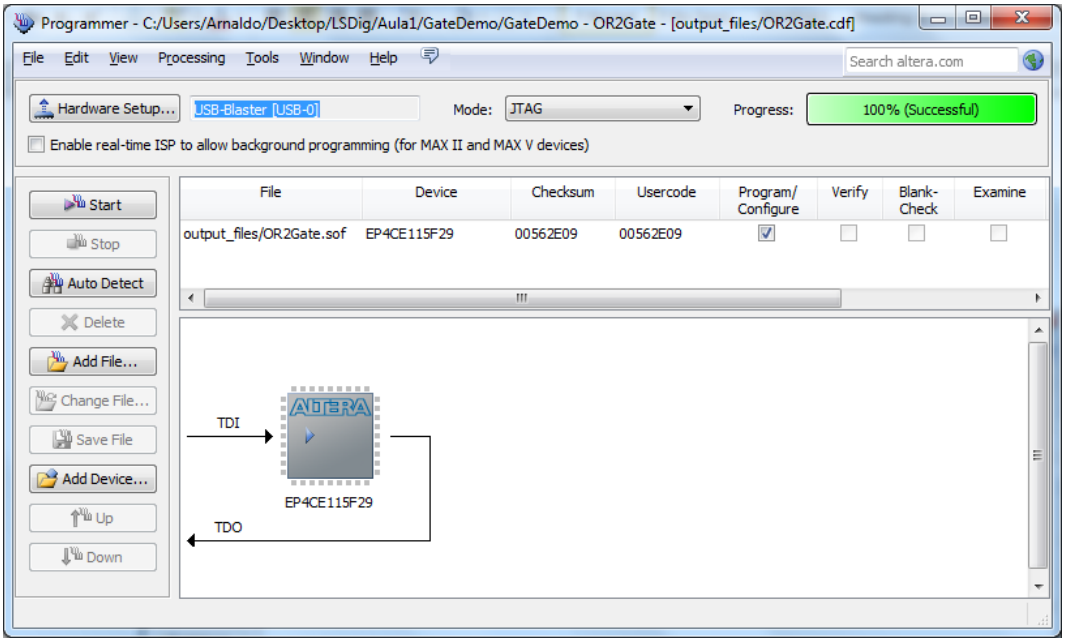


Figura 27 – Janela da aplicação após programação da FPGA.

Caso o ficheiro "output_files/OR2Gate.sof" não surja automaticamente na aplicação de programação da Figura 25, adicione-o manualmente premindo o botão *"Add File"* e selecionando-o de acordo com a Figura 28.

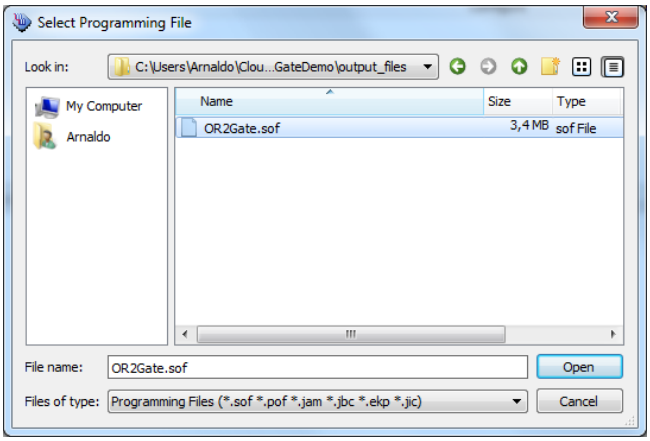


Figura 28 – Seleção do ficheiro de configuração da FPGA (caso ele já não surja automaticamente quando abre a aplicação da Figura 25.

22. Teste a porta lógica "OR" no kit de desenvolvimento aplicando diversos vetores de teste através dos interruptores usados e observando no LED o valor da saída.

23. Crie um novo ficheiro VHDL, contendo o código VHDL correspondendo a um inversor e no final grave com o nome "NOTGate.vhd" (Figura 29).

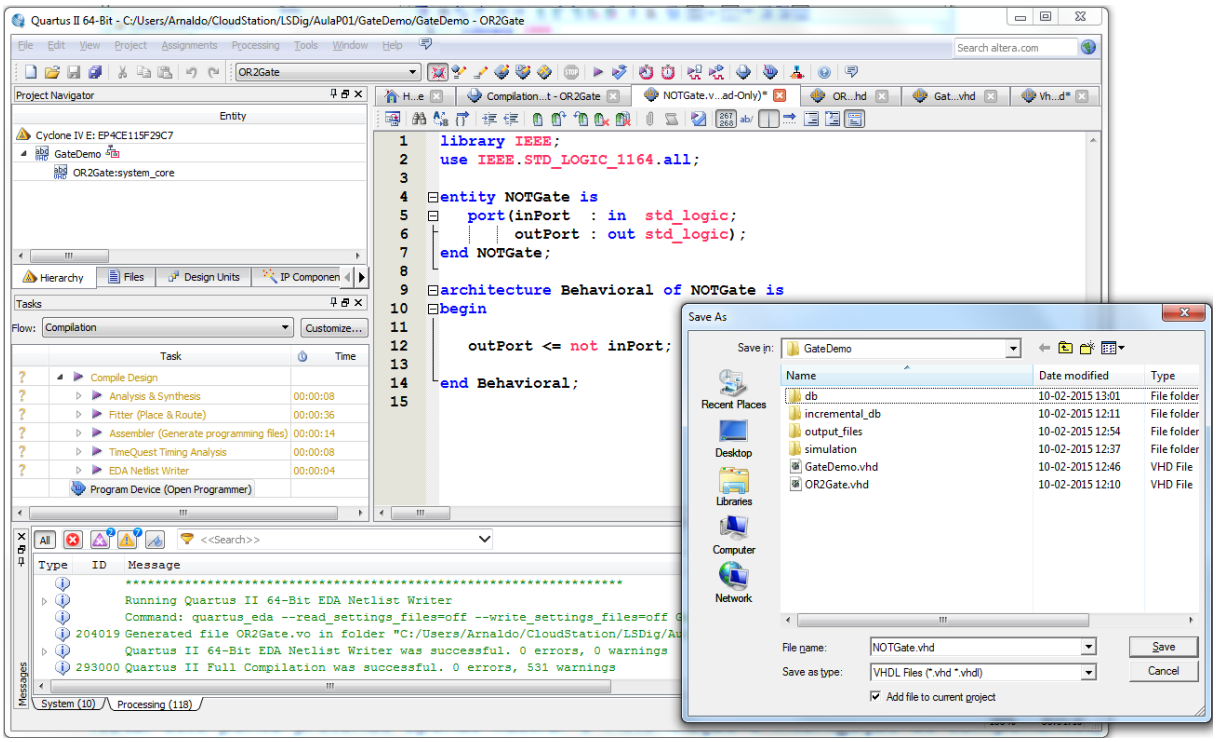


Figura 29 – Código fonte do inversor ("NOTGate.vhd").

24. Crie um novo ficheiro VHDL, contendo o código correspondente a uma porta lógica NOR de 2 entradas, construída a partir das duas portas lógicas (OR e inversor) implementadas nos pontos anteriores, instanciadas e interligadas de acordo com o código apresentado na Figura 30. No final grave o ficheiro com o nome "NOR2Gate.vhd".

**Nota:** este ponto pretende apenas ilustrar a instanciação e interligação de componentes, com base num exemplo simples, não sendo a forma mais eficaz de criar uma porta lógica NOR, uma vez que a linguagem VHDL também disponibiliza o operador "nor". No entanto,

esta forma tem a vantagem de ilustrar em VHDL, com um exemplo simples, a instanciação e interligação de componentes, de forma análoga aos métodos de captura de esquemático, mas textualmente.

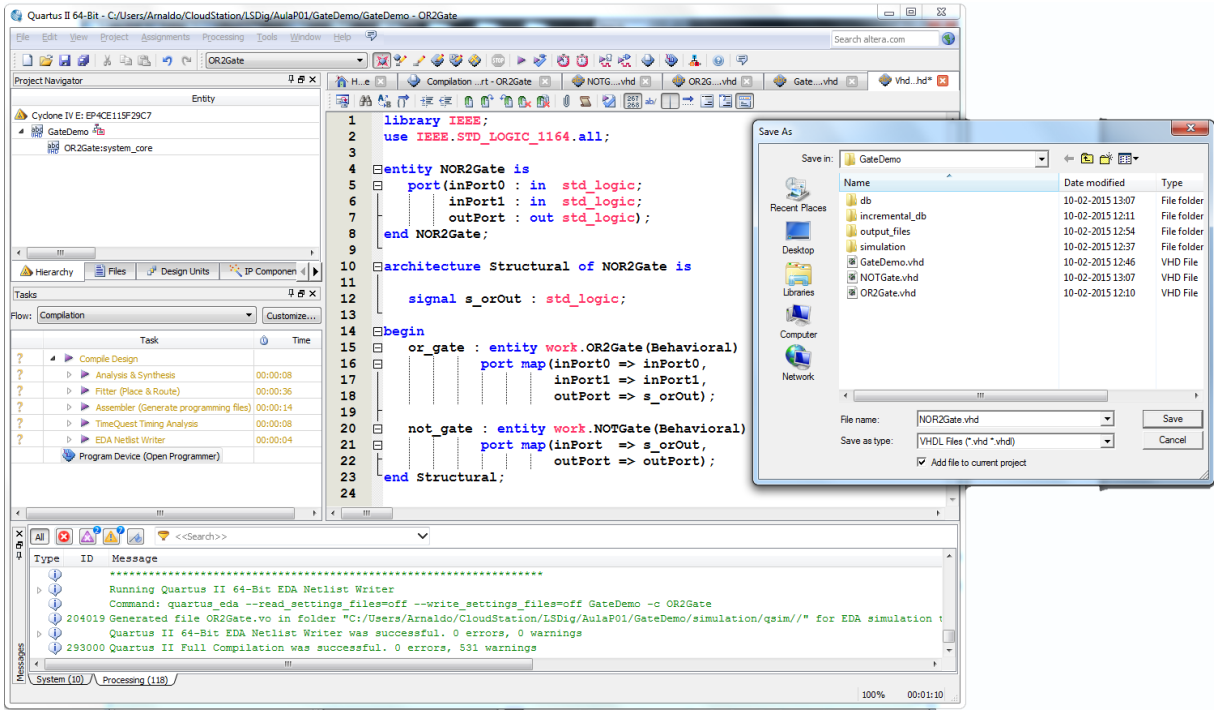


Figura 30 – Código fonte da porta lógica NOR de 2 entradas (“NOR2Gate.vhd”).

25. Altere o ficheiro “GateDemo.vhd” de forma a que seja usado como “Top Level” o componente NOR2Gate (arquitetura **`Structural`**) em vez do OR2Gate (arquitetura **`Behavioral`**) – Figuras 31 e 32.

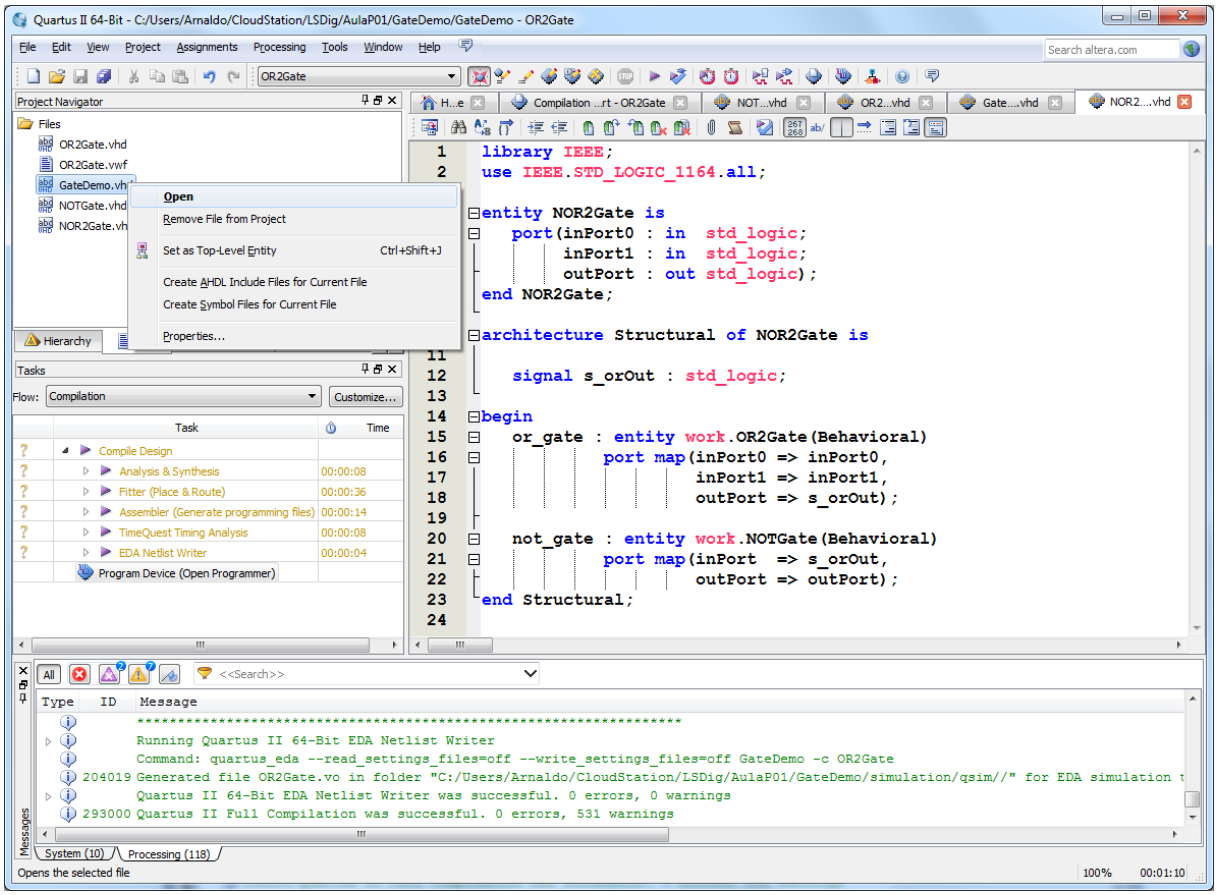


Figura 31 – Abertura do módulo *top level* "GateDemo.vhd".

26. Volte a importar o ficheiro "DE2_115.qsf" com as definições dos pinos da FPGA na placa de desenvolvimento.

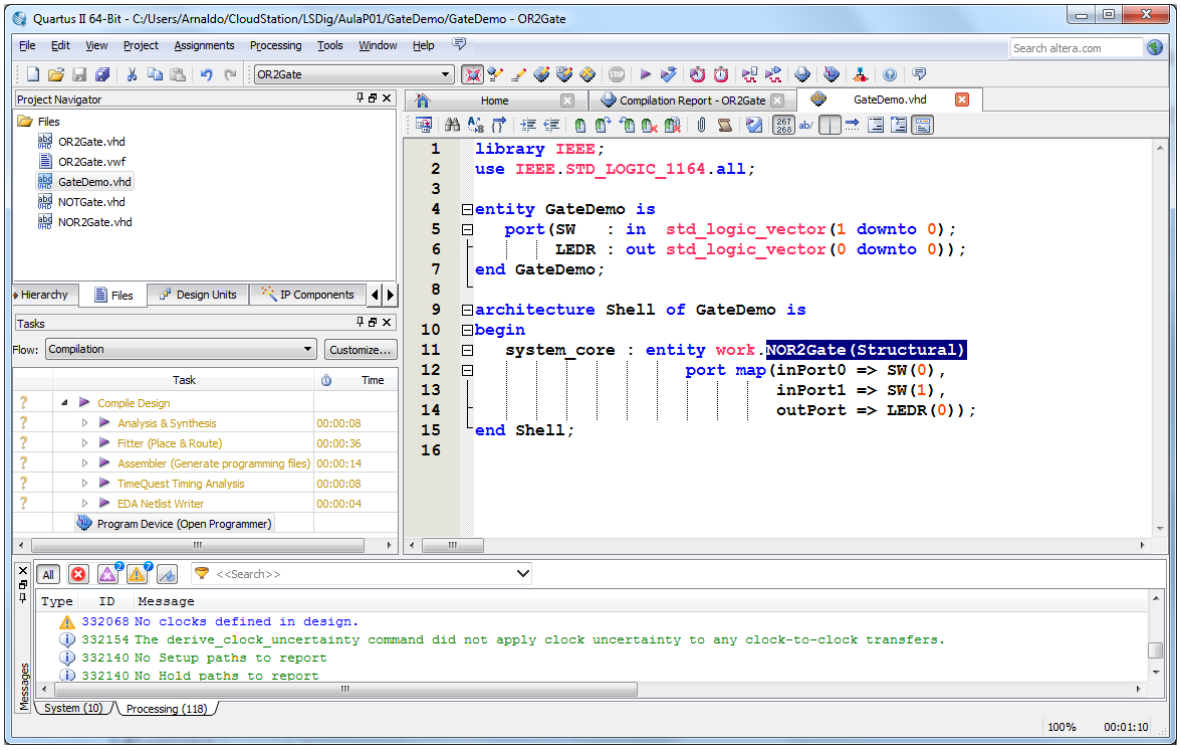


Figura 32 – Alteração do módulo *top level* "GateDemo.vhd".

27. Efetue a síntese e implementação do sistema através do comando "*Compile Design*". No final da compilação o IDE deve apresentar o aspeto da Figura 33.

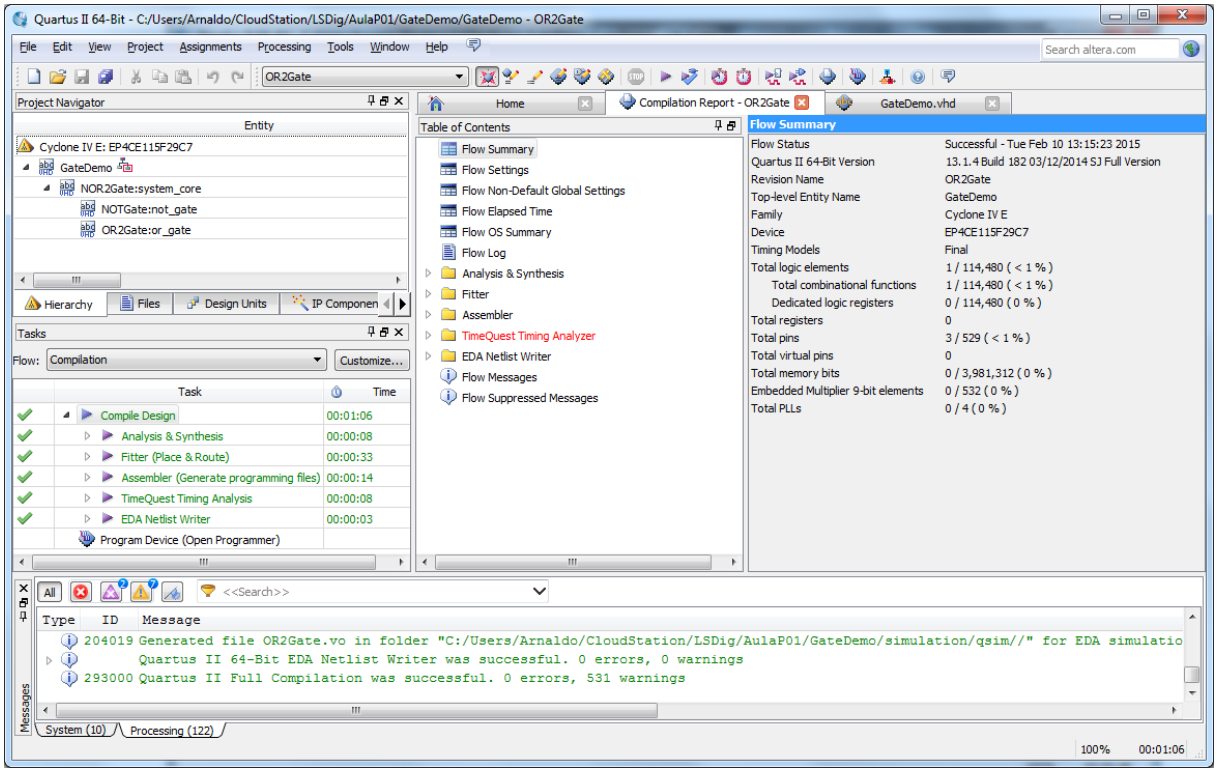


Figura 33 – Quartus II IDE após compilação completa do projeto.

28. Programe a FPGA e teste no kit o funcionamento da porta lógica NOR implementada.

29. Feche a aplicação de programação da FPGA e seguidamente o projeto.

*Parte II*

1. Crie para a FPGA do kit DE2-115 (**Cyclone IV EP4CE115F29C7**) um novo projeto, chamado "LogicDemo" e cuja entidade "Top Level" deverá chamar-se "LogicDemo".

2. Crie um novo ficheiro VHDL, chamado "LogicUnit.vhd" com o código VHDL da Figura 33, correspondente a uma unidade que realiza diversas operações lógicas.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity LogicUnit is
    port(input0  : in  std_logic;
         input1  : in  std_logic;
         invOut  : out std_logic;
         andOut  : out std_logic;
         orOut   : out std_logic;
         xorOut  : out std_logic;
         nandOut : out std_logic;
         norOut  : out std_logic);
end LogicUnit;

architecture Behavioral of LogicUnit is
begin
    invOut  <= not input0;
    andOut  <= input0 and  input1;
    orOut   <= input0 or   input1;
    xorOut  <= input0 xor  input1;
    nandOut <= input0 nand input1;
    norOut  <= input0 nor  input1;
end Behavioral;
```

Figura 34 – Código fonte de um exemplo em VHDL com operadores lógicos.

3. Crie um novo ficheiro VHDL, chamado "LogicDemo.vhd", que irá servir para instanciar a entidade "LogicUnit" e associá-la a pinos adequados da FPGA (entradas ligadas aos interruptores e saídas ligada a LEDs) do kit de desenvolvimento que vai usar para a testar.

4. Importe o ficheiro "DE2_115.qsf" com as definições dos pinos da FPGA na placa de desenvolvimento.

5. Efetue a síntese e implementação do projeto através do comando "*Compile Design*".

6. No final do processo de compilação, programe a FPGA através do comando "*Program Device*".

7. Teste o projeto no kit de desenvolvimento aplicando diversos vetores de teste através dos interruptores usados e observando nos LEDs os valores das saídas.

*Parte III*

1. Crie para a FPGA do kit DE2-115 (**Cyclone IV EP4CE115F29C7**) um novo projeto, chamado "EqCmpDemo" e cuja entidade "Top Level" deverá chamar-se "EqCmpDemo".

2. Crie um novo ficheiro VHDL, chamado "EqCmp4.vhd" com o código VHDL da Figura 34, correspondente a um comparador de igualdade de duas entradas com 4 bits.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity EqCmp4 is
    port(input0 : in  std_logic_vector(3 downto 0);
         input1 : in  std_logic_vector(3 downto 0);
         cmpOut : out std_logic);
end EqCmp4;

architecture Behavioral of EqCmp4 is
begin

    cmpOut <= '1' when (input0 = input1) else
              '0';

end Behavioral;
```

Figura 35 – Código fonte de um exemplo de um comparador em VHDL.

3. Crie um novo ficheiro VHDL, chamado "EqCmpDemo.vhd", que irá servir para instanciar o comparador e associá-lo a pinos adequados da FPGA (entradas ligadas aos interruptores e saída ligada a um LED) do kit de desenvolvimento que vai usar para o testar.

4. Importe o ficheiro "DE2_115.qsf" com as definições dos pinos da FPGA na placa de desenvolvimento.

5. Efetue a síntese e implementação do projeto através do comando "*Compile Design*".

6. No final do processo de compilação, programe a FPGA através do comando "*Program Device*".

7. Teste o comparador no kit de desenvolvimento aplicando diversos vetores de teste através dos interruptores usados e observando no LED o valor da saída.

8. Faça as alterações necessárias para efetuar a comparação de operandos de 8 bits. Teste o circuito resultante no kit.

9. Feche a aplicação de programação da FPGA e seguidamente o projeto.

*Parte IV*

1. Desenvolva um projeto que ligue/desligue uma lâmpada nas escadas duma casa (ver Figura 35). Neste caso existem dois interruptores (a - no rés-do-chão e b - no 1º andar); quando qualquer um dos interruptores é comutado, comuta também a iluminação das escadas. Inicialmente, quando os dois interruptores estão desativados, a luz encontra-se apagada.

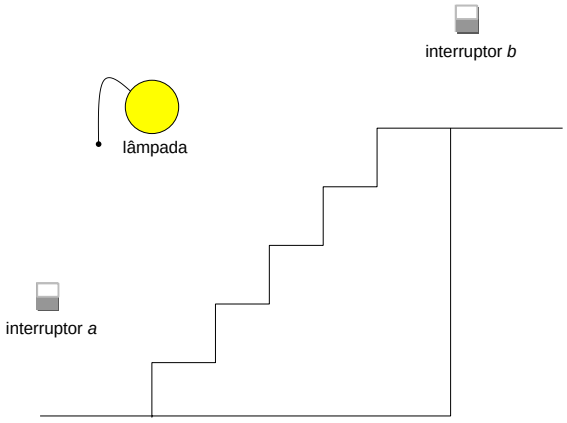


Figura 36 – Posicionamento relativo dos interruptores e da lâmpada.

a) Identifique as entradas e saídas do circuito.
b) Escreva a equação lógica da saída que comanda a lâmpada, L(a,b), admitindo que L=1 corresponde a luz acesa.
c) Implemente e teste o circuito usando dois interruptores e um LED do kit DE2-115.

2. Uma sala tem 3 portas de acesso. Junto de cada porta está um interruptor (a, b, c) capaz de ligar ou desligar a luz. Quando qualquer um dos interruptores é comutado, comuta também a iluminação da sala. Admite-se que a luz está apagada quando todos os interruptores estão desativados.

a) Escreva a equação lógica da iluminação, L(a,b,c), admitindo que L=1 corresponde a luz acesa.
b) Implemente o circuito e teste-o usando três interruptores e um LED do kit DE2-115.
c) Que alterações devem ser feitas no circuito de modo a incluir mais uma porta?