# Circuitos Digitais II - 6882

André Barbosa Verona
Nardênio Almeida Martins

Universidade Estadual de Maringá
Departamento de Informática

Bacharelado em Ciência da Computação

# Aula de Hoje

Projeto e Simulação de um circuito multiplexador 2 X 1:

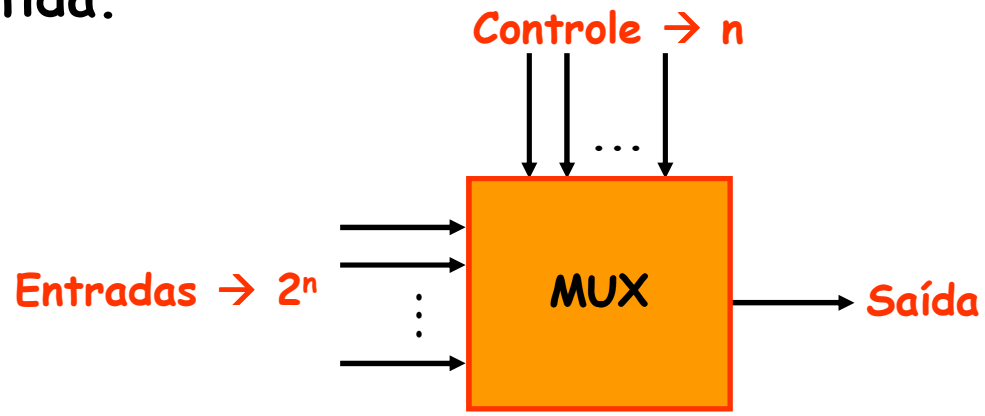
- Modelagem ou arquitetura por fluxo de dados
- Modelagem ou arquitetura comportamental
- Modelagem ou arquitetura estrutural

# HDL – Linguagem de Descrição de Hardware

## Multiplexador

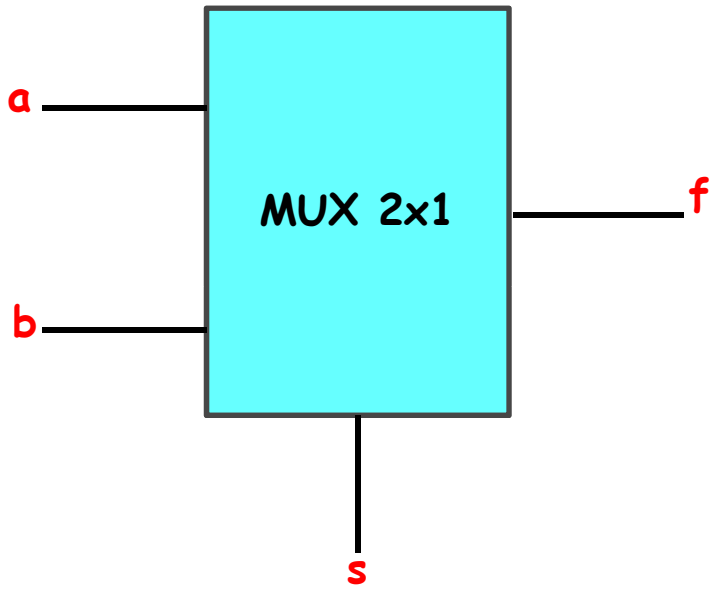
**Multiplexador ou Seletor de Dados:** É um circuito lógico que tem diversas entradas e apenas uma saída. MUX seleciona uma única entrada para transmitir para a saída.

**Entradas de Controle:** permitem selecionar a entrada a ser transmitida.



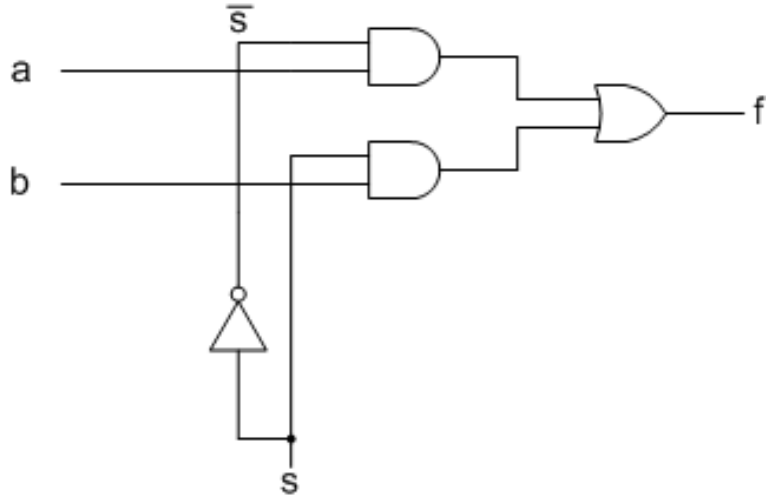
# HDL – Linguagem de Descrição de Hardware

## Multiplexador 2 X 1




TV

| s | f |
|---|---|
| 0 | a |
| 1 | b |




$$f = \overline{s}.a + s.b$$

# HDL – Linguagem de Descrição de Hardware

## Estrutura Básica de um Código em VHDL

| | |
|---|---|
| **LIBRARY IEEE;**<br>**USE IEEE.STD_LOGIC_1164.all;**<br>**USE IEEE.STD_LOGIC_UNSIGNED.all;** | LIBRARY (PACOTES) |
| **ENTITY** exemplo **IS**<br>**PORT** (<br><descrição dos pinos de I/O><br>);<br>**END** exemplo; | ENTITY (PINOS DE I/O) |
| **ARCHITECTURE** teste **OF** exemplo **IS**<br>**BEGIN**<br>...<br>**END** teste; | ARCHITECTURE<br>(ARQUITETURA) |

# HDL – Linguagem de Descrição de Hardware

## Multiplexador 2 X 1

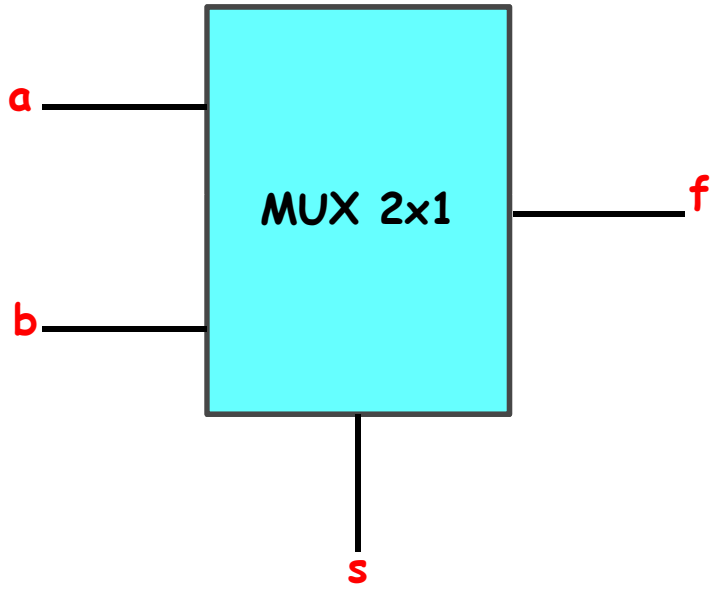
### Modelagem ou arquitetura por fluxo de dados em VHDL

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY mux2_fd IS
     PORT (a, b : IN BIT;
                s  : IN BIT;
                f   : OUT BIT);
END mux2_fd;
ARCHITECTURE fluxo_de_dados  OF mux2_fd IS
BEGIN
          f <= ((NOT s ) AND a) OR (s AND b);
END fluxo_de_dados;
```

# HDL – Linguagem de Descrição de Hardware

## Multiplexador 2 X 1

### Modelagem comportamental em VHDL usando IF THEN ELSE




TV

| s | f |
|---|---|
| 0 | a |
| 1 | b |

**Condição:**

Se a variável de entrada **s** for igual a **0** então a variável de saída **f** será igual a **a**, senão a variável de saída **f** será igual a **b**.

# HDL – Linguagem de Descrição de Hardware

## Multiplexador 2 X 1

### Modelagem comportamental em VHDL usando IF THEN ELSE

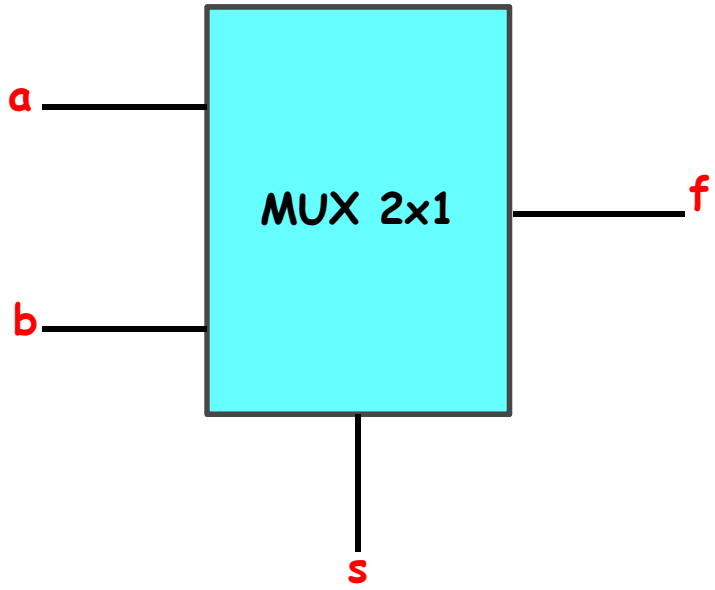
Declaração de um processo

```
. . .
ARCHITECTURE nome_da_arquitetura OF nome_da_entidade IS
BEGIN
        PROCESS (lista_de_sensibilidade)
        BEGIN
                . . .
                comandos;
                . . .
        END PROCESS;
END nome_da_arquitetura;
```

# HDL – Linguagem de Descrição de Hardware

## Multiplexador 2 X 1

### Modelagem comportamental em VHDL usando WHEN ELSE




TV

| s | f |
|---|---|
| 0 | a |
| 1 | b |

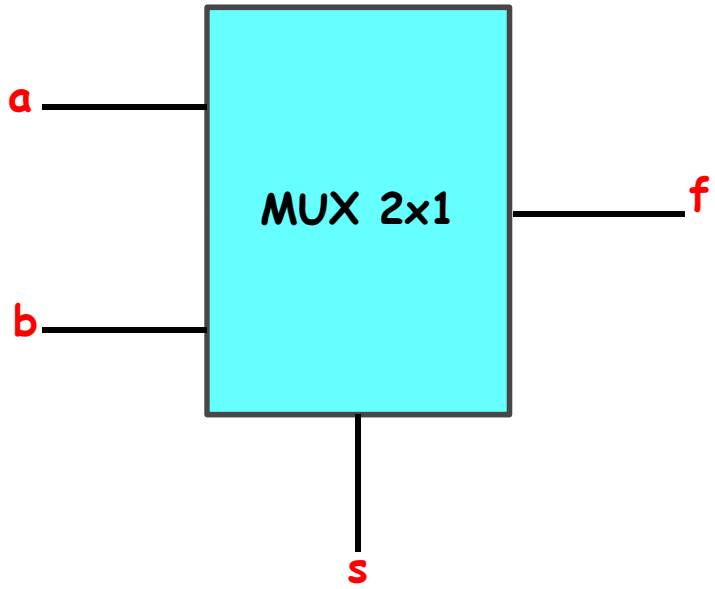
**Condição:**

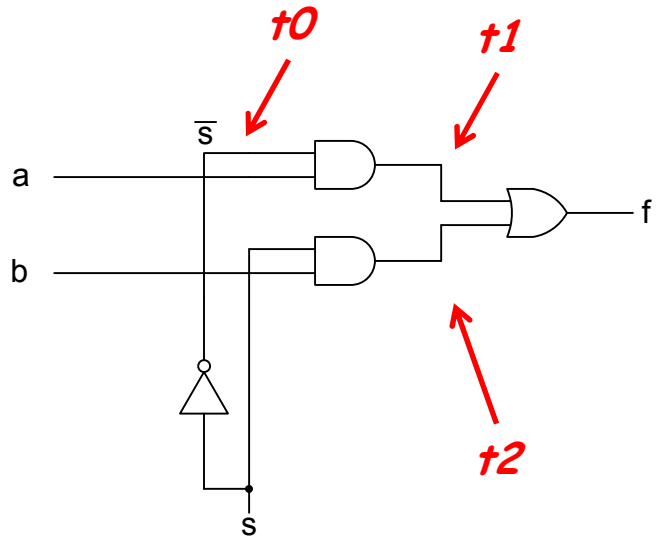
A variável de saída **f** será igual a **a** quando a variável de entrada **s** for igual a **0**, senão a variável de saída **f** será igual a **b**.

# HDL – Linguagem de Descrição de Hardware

## Multiplexador 2 X 1

### Modelagem estrutural em VHDL



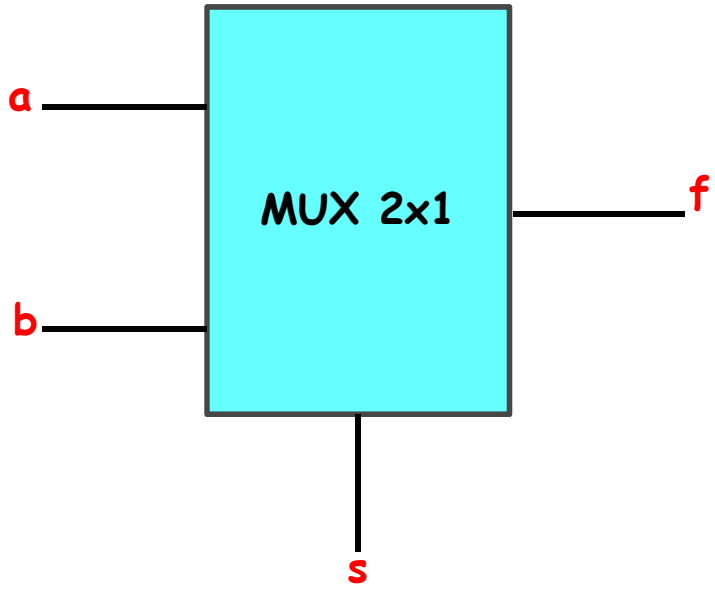



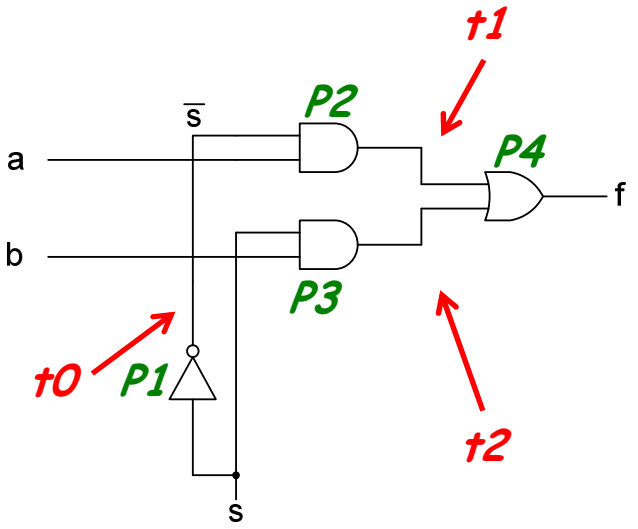

- A descrição estrutural especifica separadamente a função de cada porta no multiplexador.
- Essa descrição é baseada na expressão booleana do circuito.
- Uso de sinais → t0, t1 e t2.

# HDL – Linguagem de Descrição de Hardware

## Multiplexador 2 X 1

### Modelagem estrutural em VHDL usando componentes



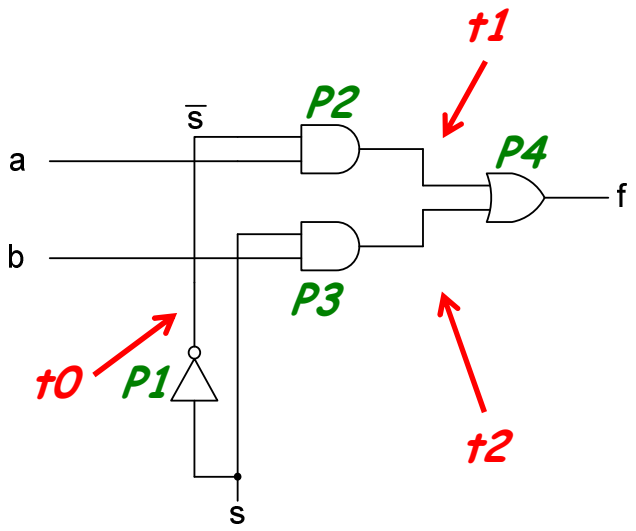

- A descrição estrutural especifica separadamente a função de cada porta no multiplexador.
- Essa descrição é baseada na expressão booleana do circuito.

# HDL – Linguagem de Descrição de Hardware

## Multiplexador 2 X 1

### Modelagem estrutural em VHDL usando componentes

- Criação dos componentes (portas NOT, AND e OR) com entidade de projeto (declaração da entidade e arquitetura da entidade).
- Declaração dos componentes na arquitetura da entidade da entidade de projeto → declaração entre as palavras reservadas IS e BEGIN.
- Instanciação dos componentes usando a palavra reservada PORT MAP na arquitetura da entidade da entidade de projeto → instanciação entre as palavras reservadas BEGIN e END.



# HDL – Linguagem de Descrição de Hardware

## Multiplexador 2 X 1

### Modelagem estrutural em VHDL usando componentes

- Declaração e instanciação de um componente

```
. . .
ARCHITECTURE nome_da_arquitetura OF nome_da_entidade IS
COMPONENT nome_do_componente                    -- declaracao do componente
        PORT (lista_de_porta_de_interface);
END COMPONENT;
SIGNAL  lista_de_sinal : tipo_de_dado;
BEGIN
        nome_da_instanciacao : nome_componente PORT MAP
       (lista_de_associacao_de_porta);          -- instanciacao do componente
END nome_da_arquitetura;
```

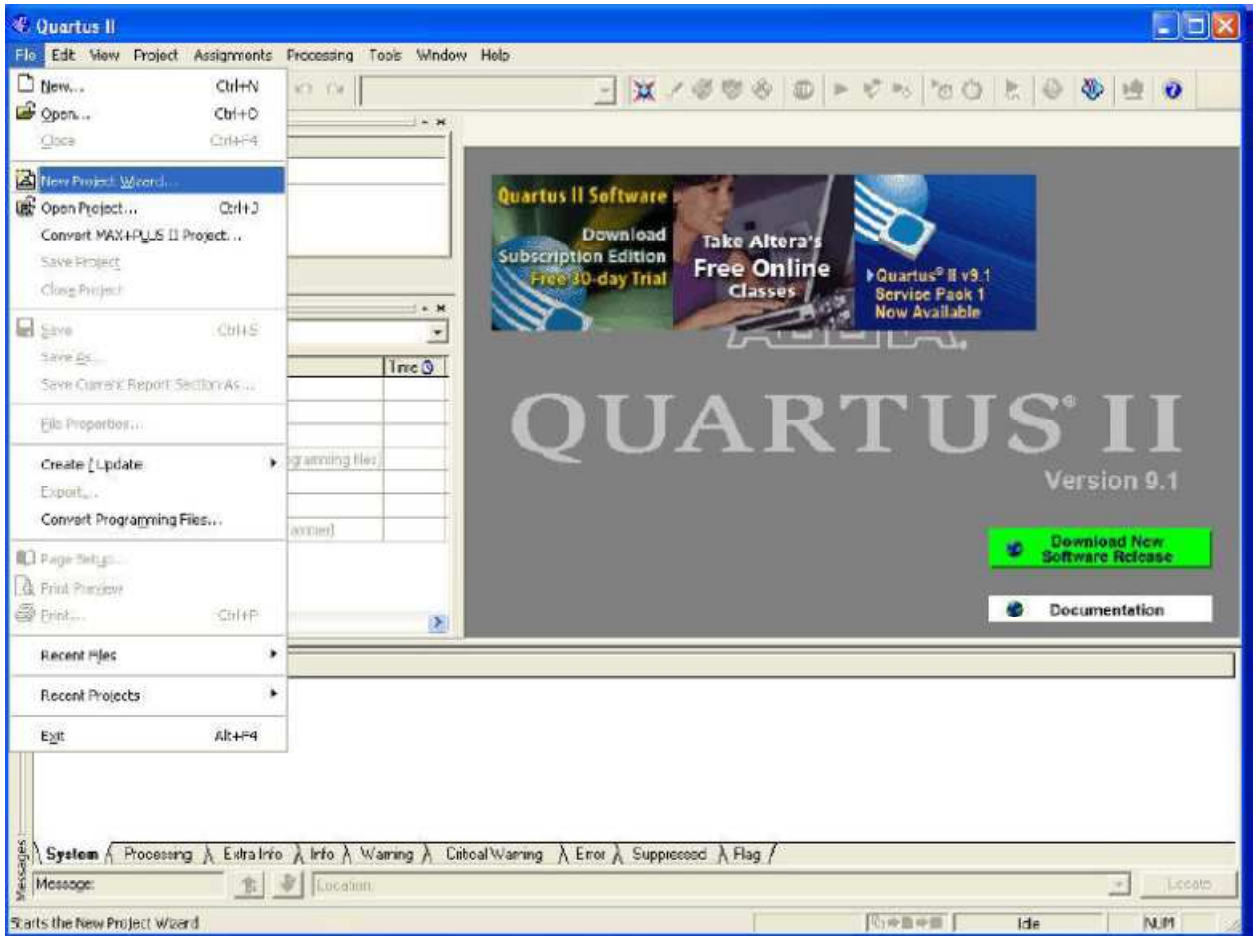
- Nota: A declaração do componente é similar à declaração de entidade.

# Software Quartus II

1. Crie diretório ou pasta "work" na área de trabalho.
2. Crie os seguintes subdiretórios dentro do diretório "work":
   a) "mux2_fd"
   b) "mux2_com"
   c) "mux2_com1"
   d) "mux2_est"
   e) "mux2_est1" (uso de componente)
3. Inicialize o Software Quartus II

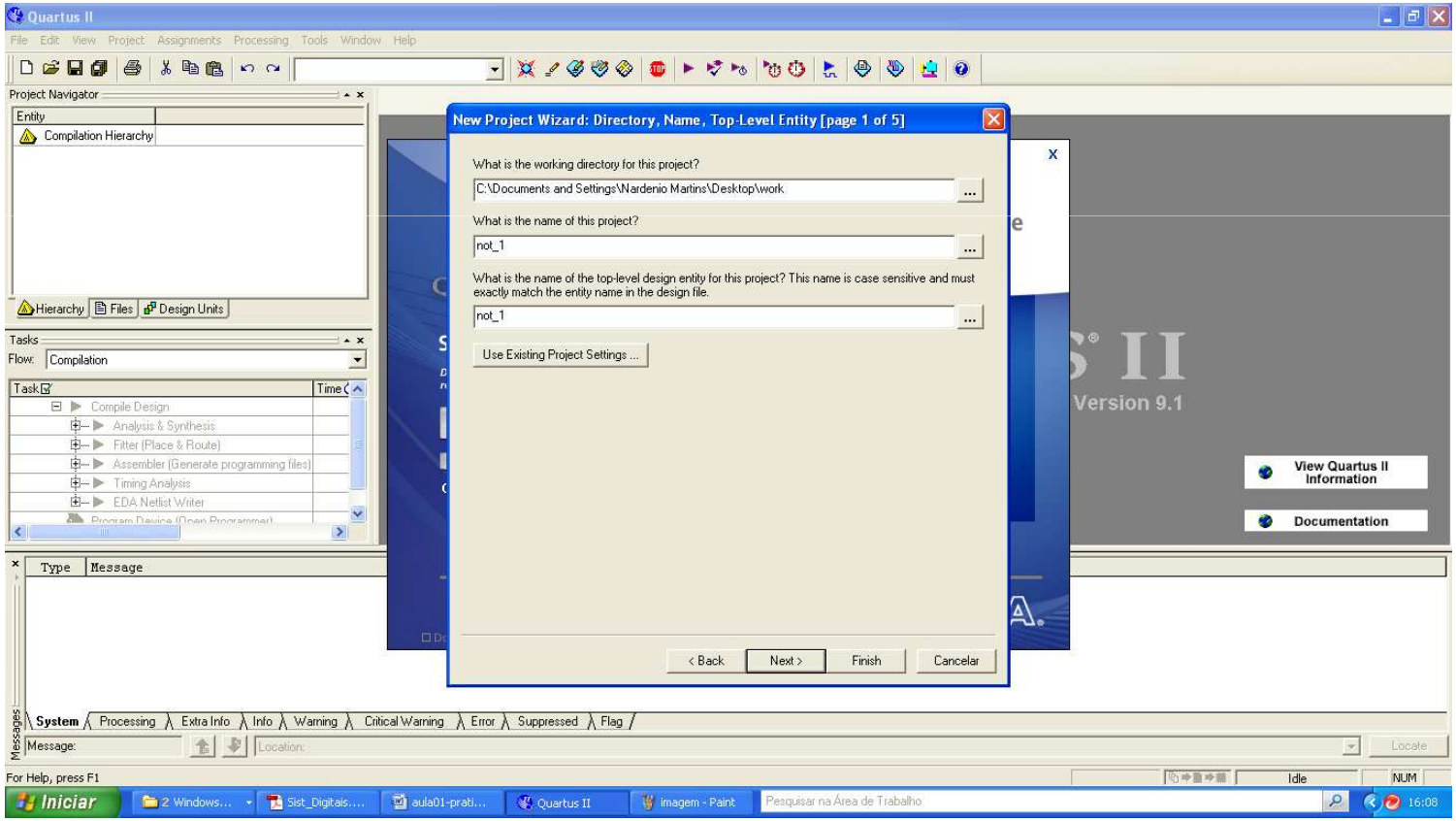
# Software Quartus II

4. Crie um novo projeto: selecione "File > New Project Wizard"



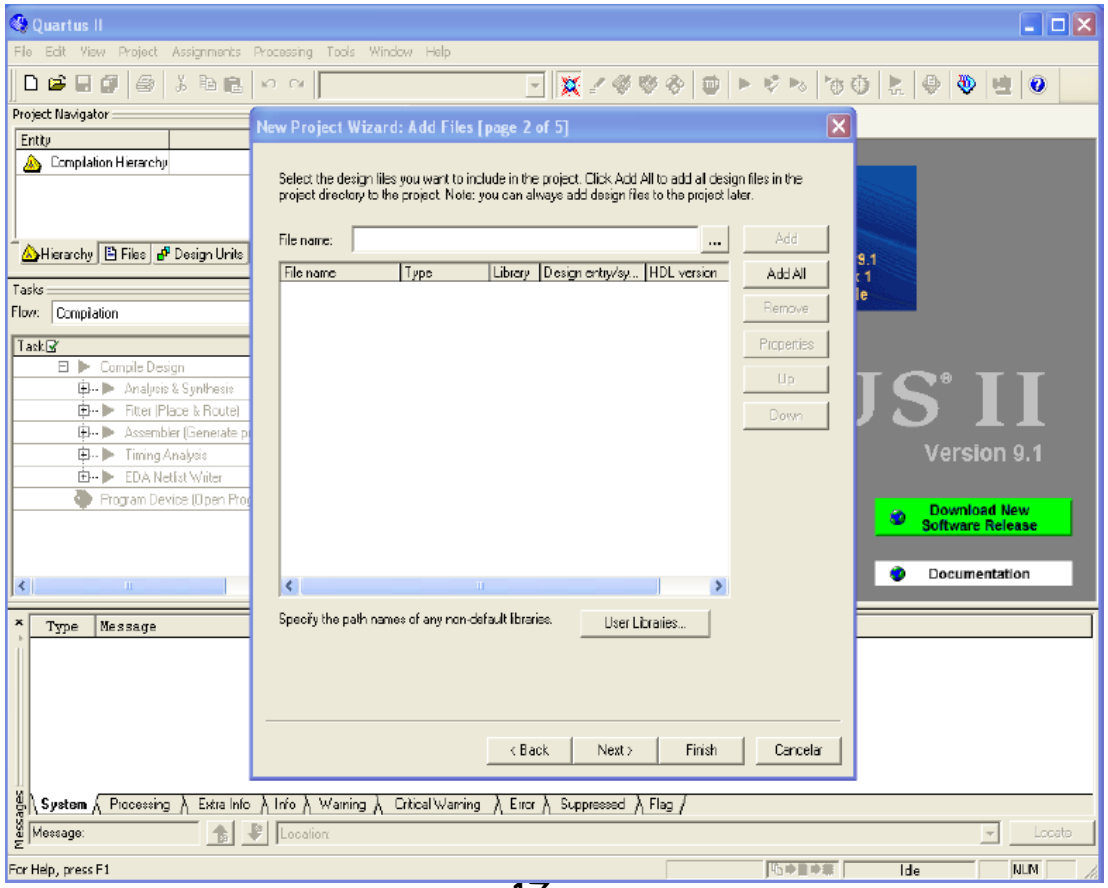
# Software Quartus II

5. Na primeira linha da janela, insira o caminho e o nome do diretório do projeto → "work". Na segunda linha insira o nome do projeto → "mux2_est1".



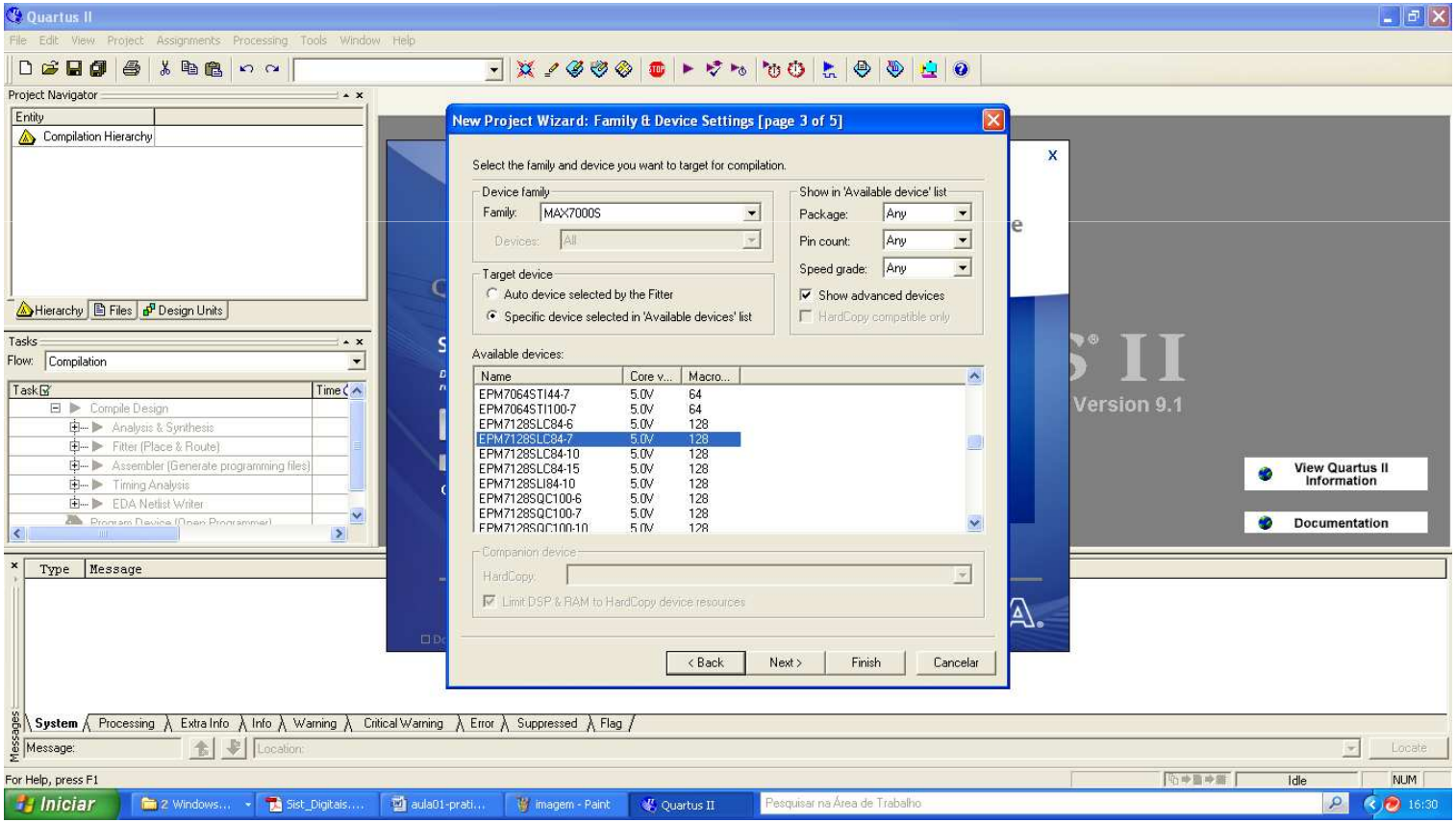
# Software Quartus II

6. Pressione "Next". O projetista pode incluir arquivos de outros projetos, ou mesmo aqueles que estão nas "*Libraries*" do software Quartus II.



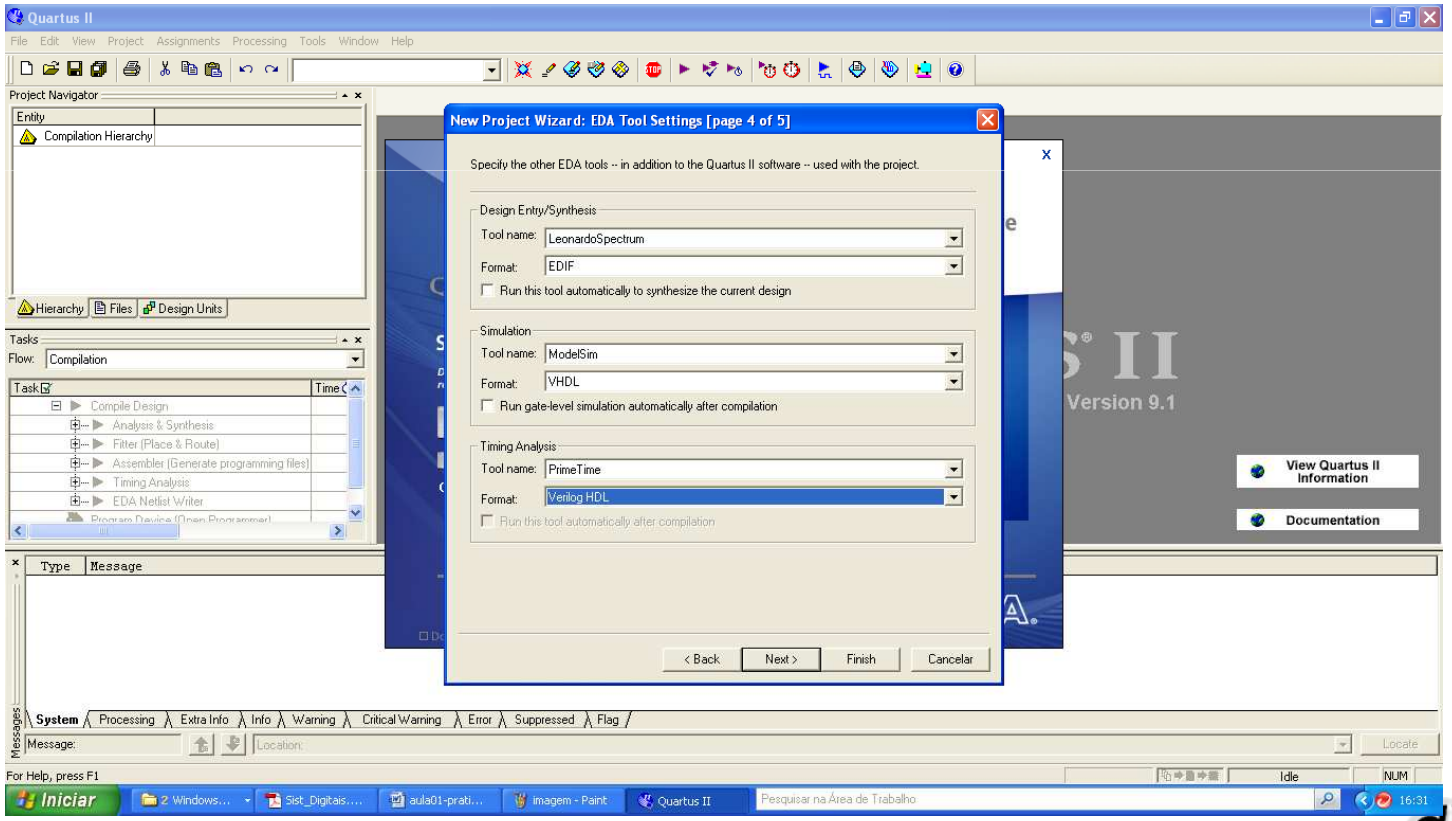
# Software Quartus II

7. Selecione o dispositivo lógico programável a ser utilizado. Neste caso é usado o CPLD da família "MAX7000S", denominado "EPM7128SLC84-7".



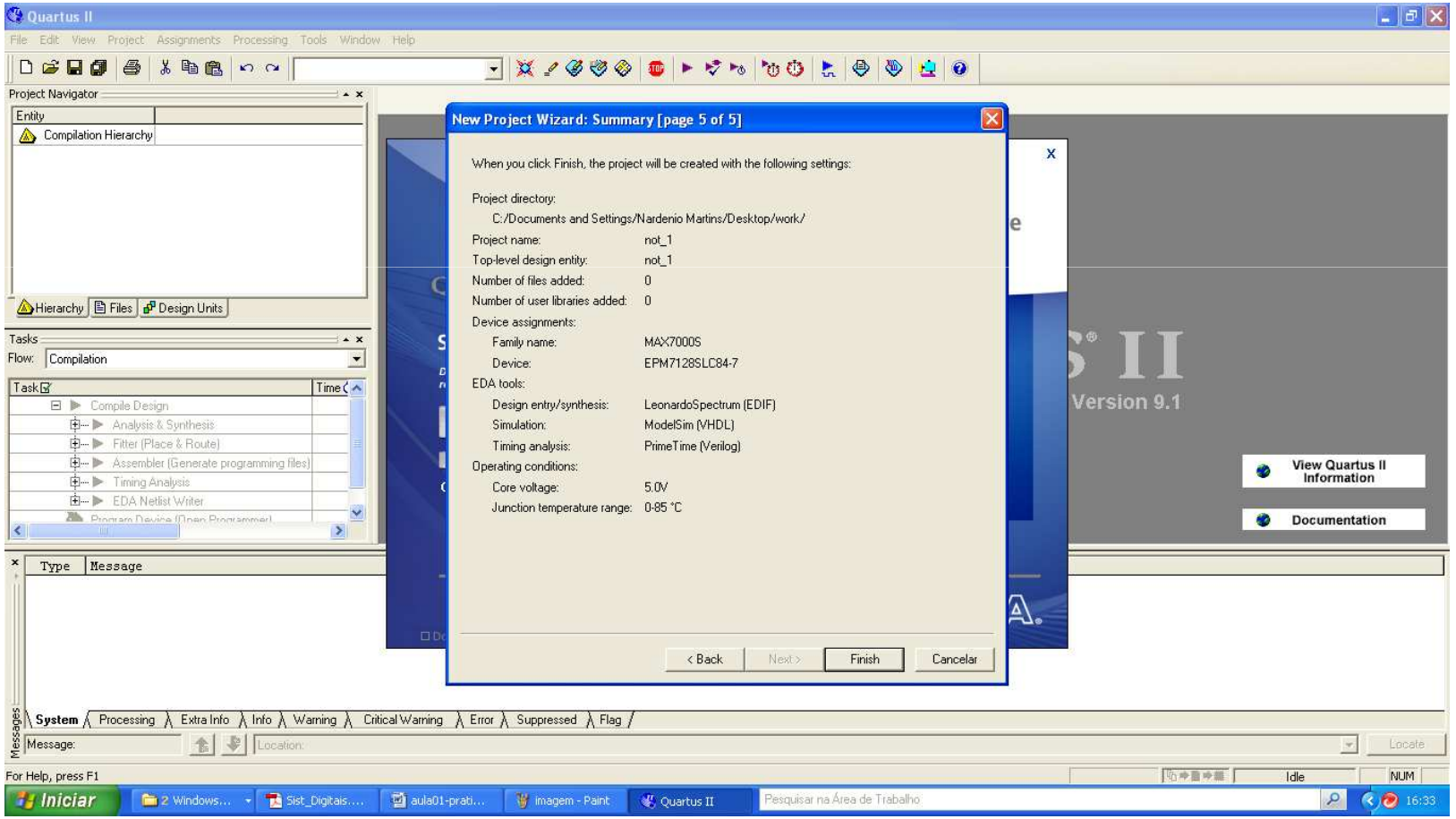
# Software Quartus II

8. O próximo passo permite a adição de outras ferramentas como "LeonardoSpectrum" e "EDIF", "ModelSim" e "VHDL", "PrimeTime" e "Verilog HDL" que possibilita a interação entre FPGA e ASIC.



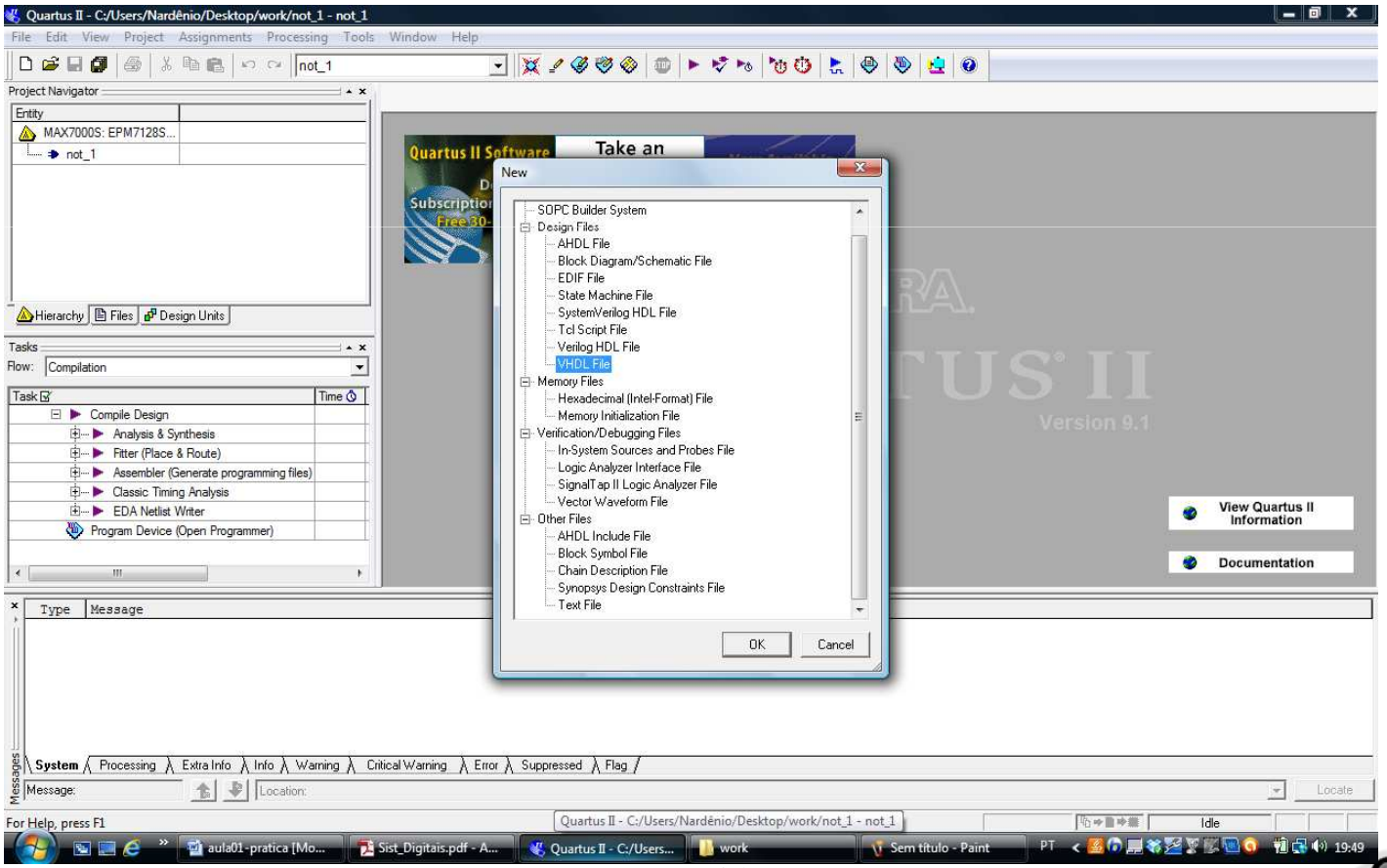
# Software Quartus II

9. O último passo apresenta um resumo do projeto a ser executado. Posteriormente, clique em "Finish".



# Software Quartus II

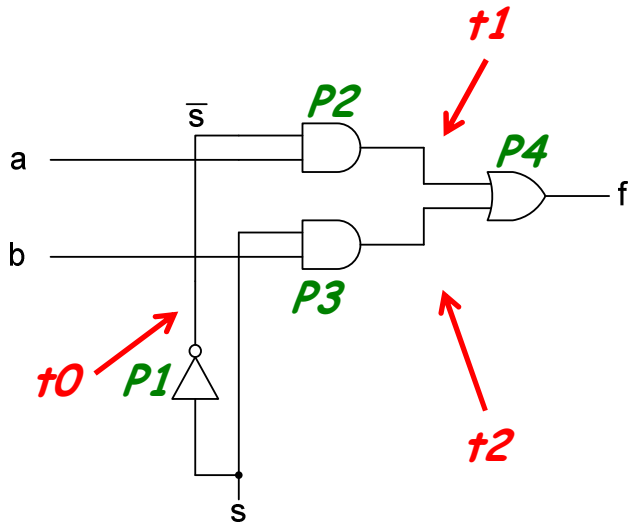
10. Defina o modo a ser utilizado para desenvolver o projeto: AHDL, VHDL ou Block Diagram/Schematic File. Selecione "File > New" e escolha "VHDL file".



# Software Quartus II

11. Escreva o código em VHDL da função lógica NOT e salve o código em VHDL com extensão "not_1.vhd" na pasta ou subdiretório "mux2_est1". Está criado o componente "not_1".

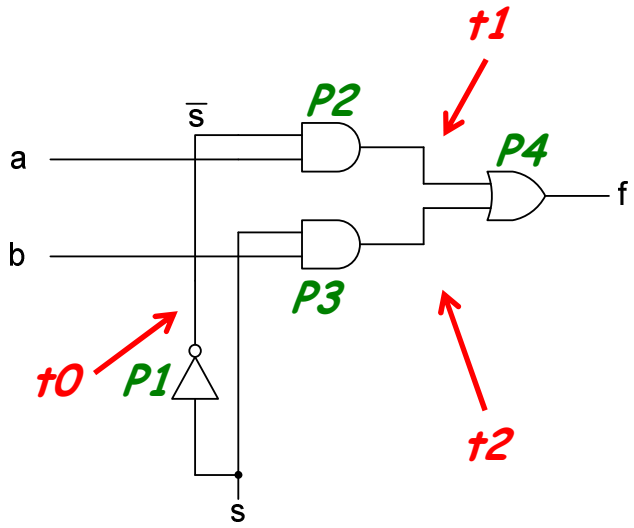
```
ENTITY not_1 IS
    PORT (x : IN BIT;
          z : OUT BIT);
END not_1;
ARCHITECTURE logica OF not_1 IS
BEGIN
        z <= NOT x;
END logica;
```



# Software Quartus II

12. Selecione "File > New" e escolha "VHDL file". Escreva o código em VHDL da função lógica AND e salve o código em VHDL com extensão "and_2.vhd" na pasta ou subdiretório "mux2_est1". Está criado o componente "and_2".

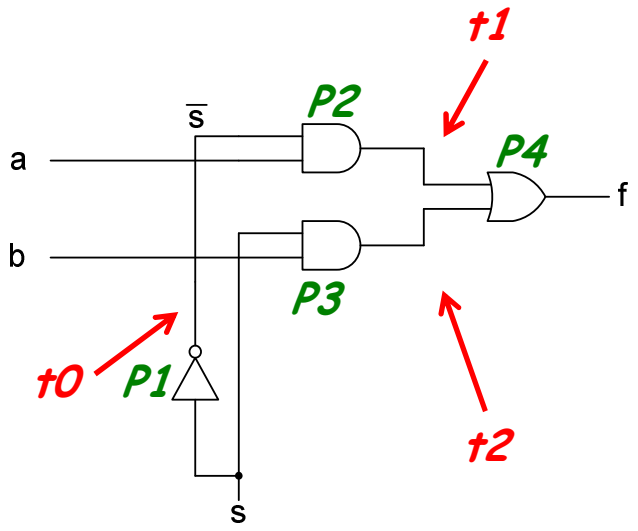
```
ENTITY and_2 IS
        PORT (x, y : IN BIT;
                     z : OUT BIT);
END and_2;
ARCHITECTURE logica OF and_2 IS
BEGIN
             z <= x AND y;
END logica;
```



# Software Quartus II

13. Selecione "File > New" e escolha "VHDL file". Escreva o código em VHDL da função lógica OR e salve o código em VHDL com extensão "or_2.vhd" na pasta ou subdiretório "mux2_est1". Está criado o componente "or_2".

```
ENTITY or_2 IS
        PORT (x, y : IN BIT;
                     z : OUT BIT);
END or_2;
ARCHITECTURE logica OF or_2 IS
BEGIN
            z <= x OR y;
END logica;
```



# Software Quartus II

14. Selecione "File > New" e escolha "VHDL file". Escreva o código em VHDL do próximo slide. Neste código ocorre a declaração dos componentes criados e dos sinais (declaração entre IS e BEGIN da arquitetura da entidade), bem como a instanciação destes componentes criados (instanciação entre BEGIN e END da arquitetura da entidade).

# Software Quartus II

## Modelagem estrutural em VHDL usando componentes

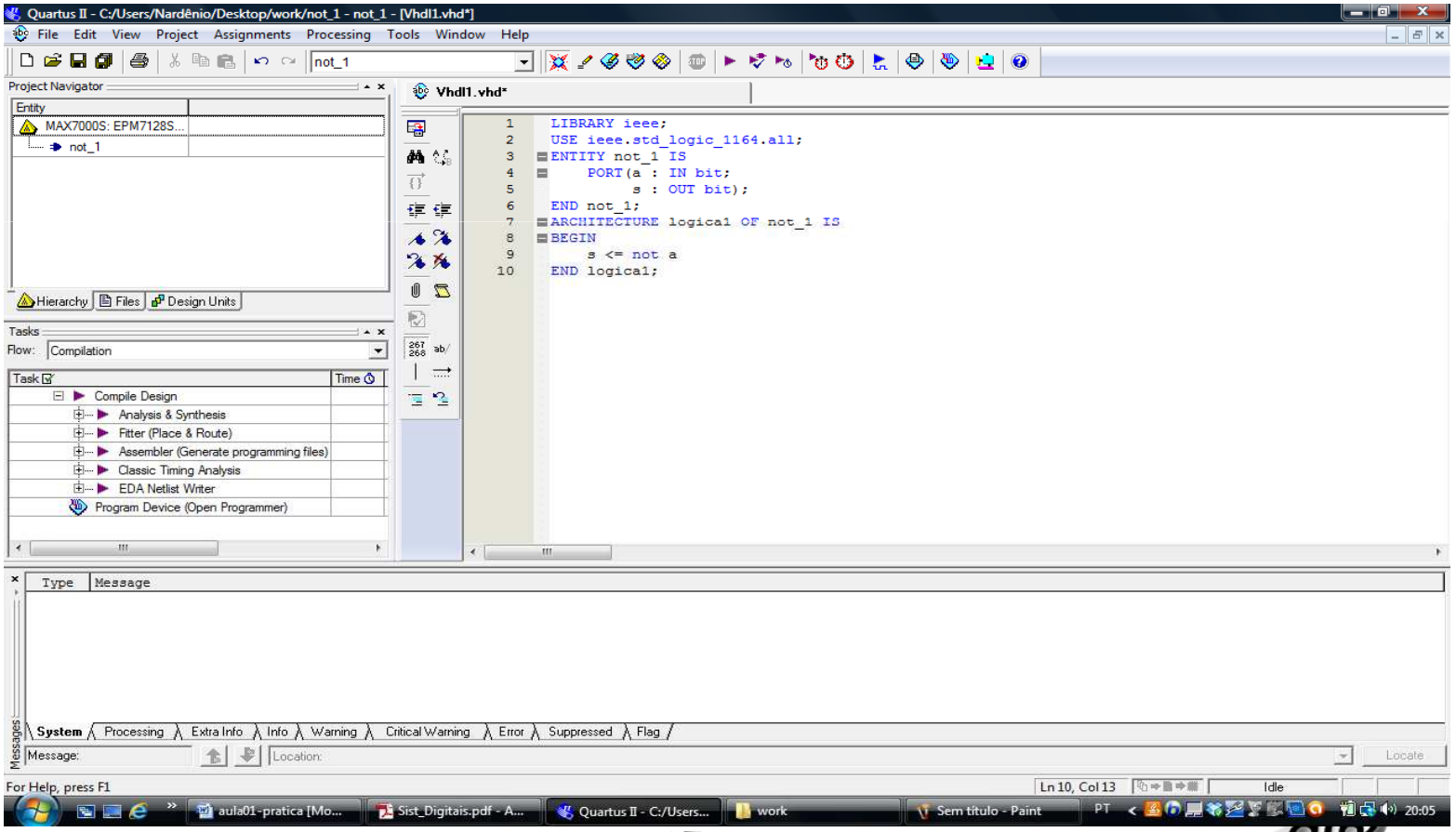
```
ENTITY mux2_est1 IS
    PORT (a, b : IN BIT;
          s : IN BIT;
          f : OUT BIT);
END mux2_est1;
ARCHITECTURE estrutural OF mux2_est1 IS
COMPONENT not1
    PORT (x : IN BIT;
          z : OUT BIT);
END component;
COMPONENT and2
    PORT (x, y : IN BIT;
          z : OUT BIT);
END component;                    -- continuacao
```

```
COMPONENT or2      -- continuacao
    PORT (x, y : IN BIT;
          z : OUT BIT);
END component;

SIGNAL t0, t1, t2 : BIT;
BEGIN
    P1: not1 PORT MAP (s, t0);
    P2: and2 PORT MAP (t0, a, t1);
    P3: and2 PORT MAP (s, b, t2);
    P4:  or2 PORT MAP (t1, t2, f);
END estrutural;
```

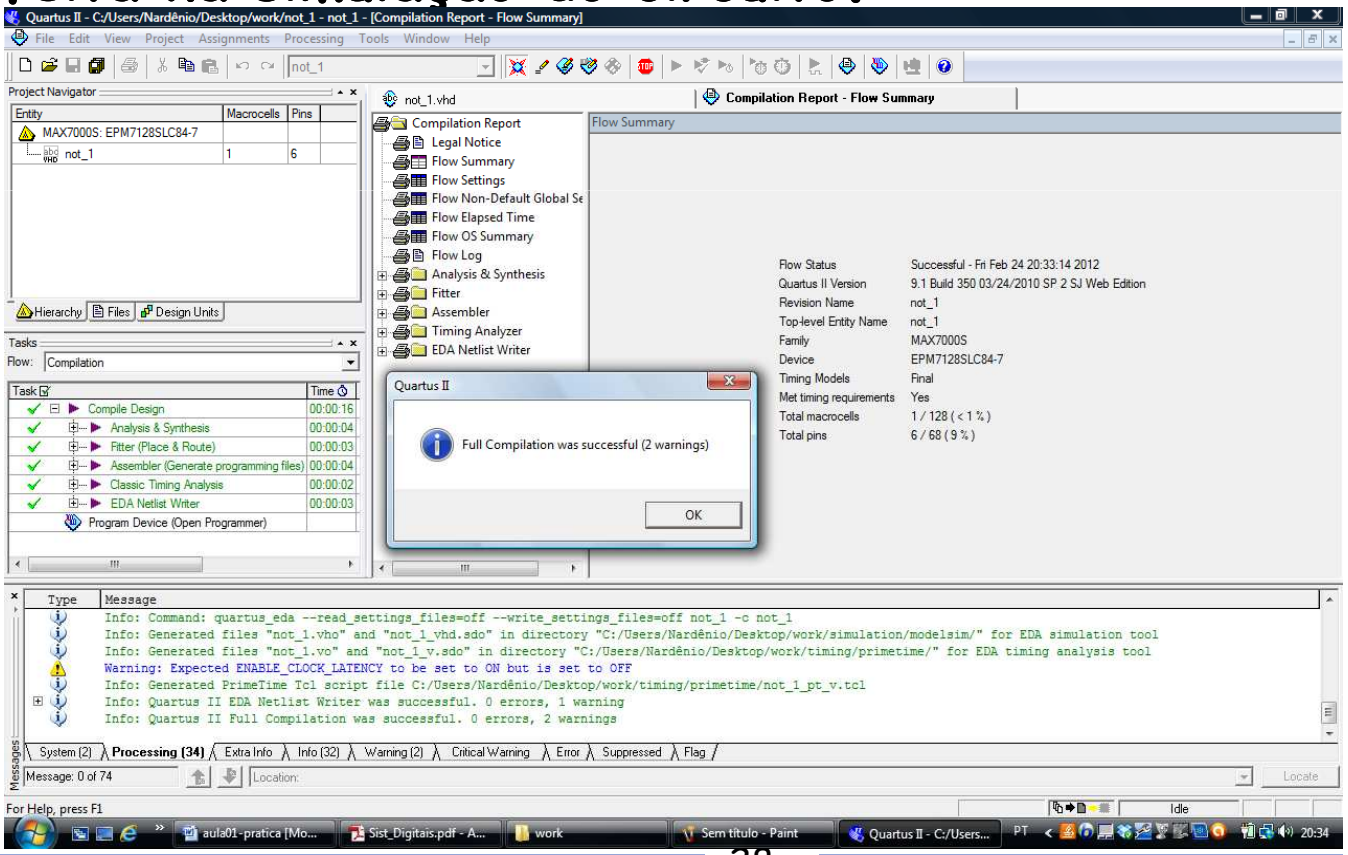
# Software Quartus II

15. Salve o código em VHDL com extensão "mux2_est1.vhd" na pasta ou subdiretório "mux2_est1".



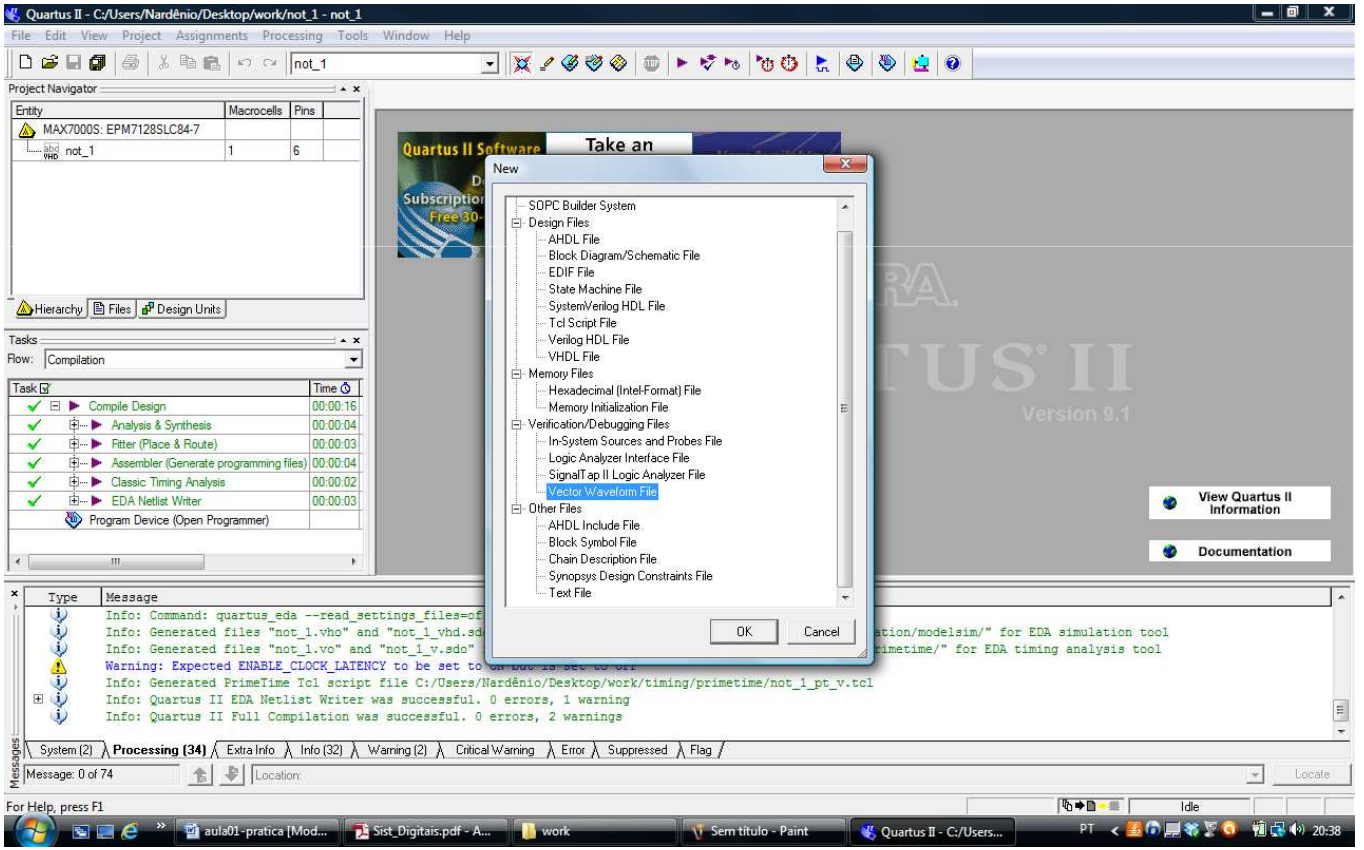
# Software Quartus II

16. Compilação: "Processing > Start Compilation". A compilação é a verificação da construção. Nesta etapa, erros lógicos não são detectados. Esta verificação é feita na simulação do circuito.



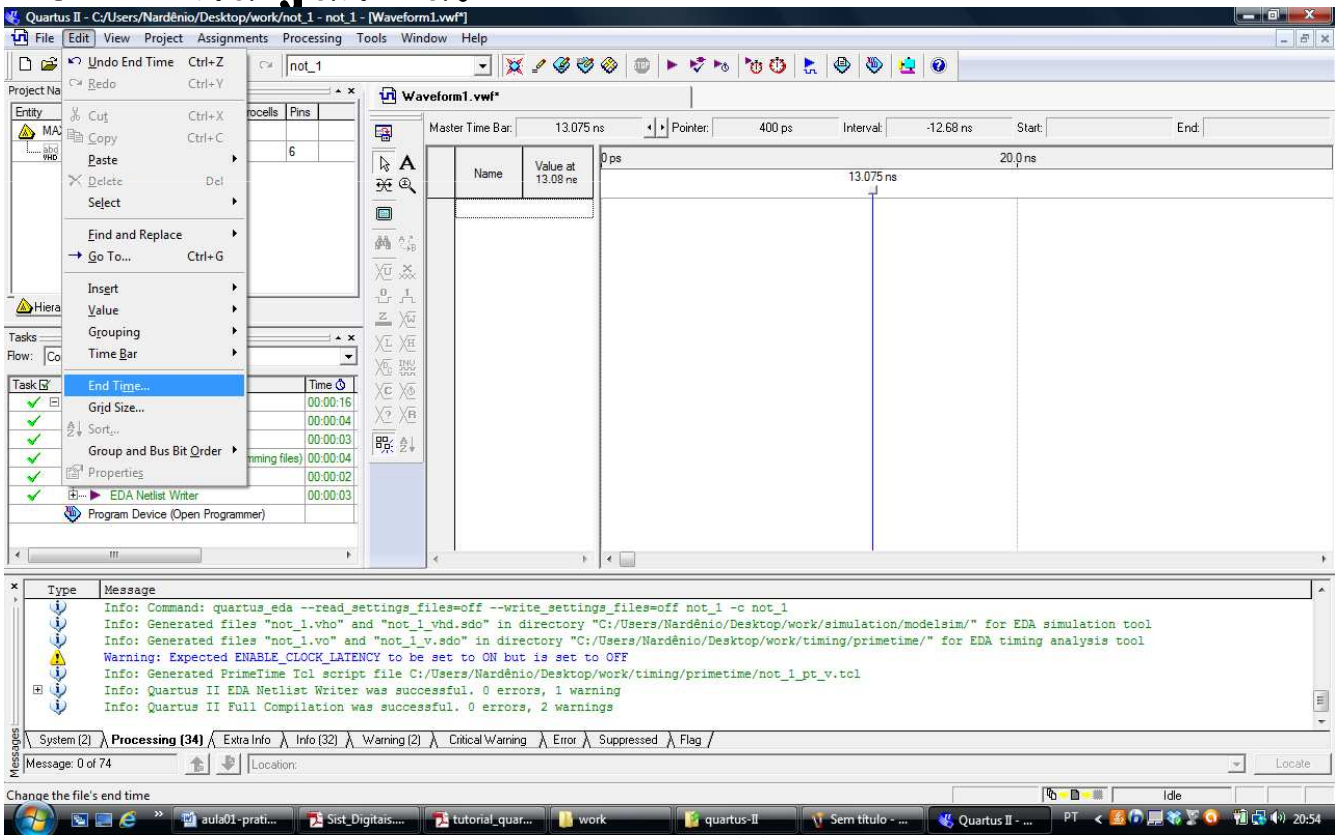
# Software Quartus II

17. A verificação de erros lógicos é feita na simulação do circuito. Para isto selecione "File > New" e escolha "Vector Waveform File".



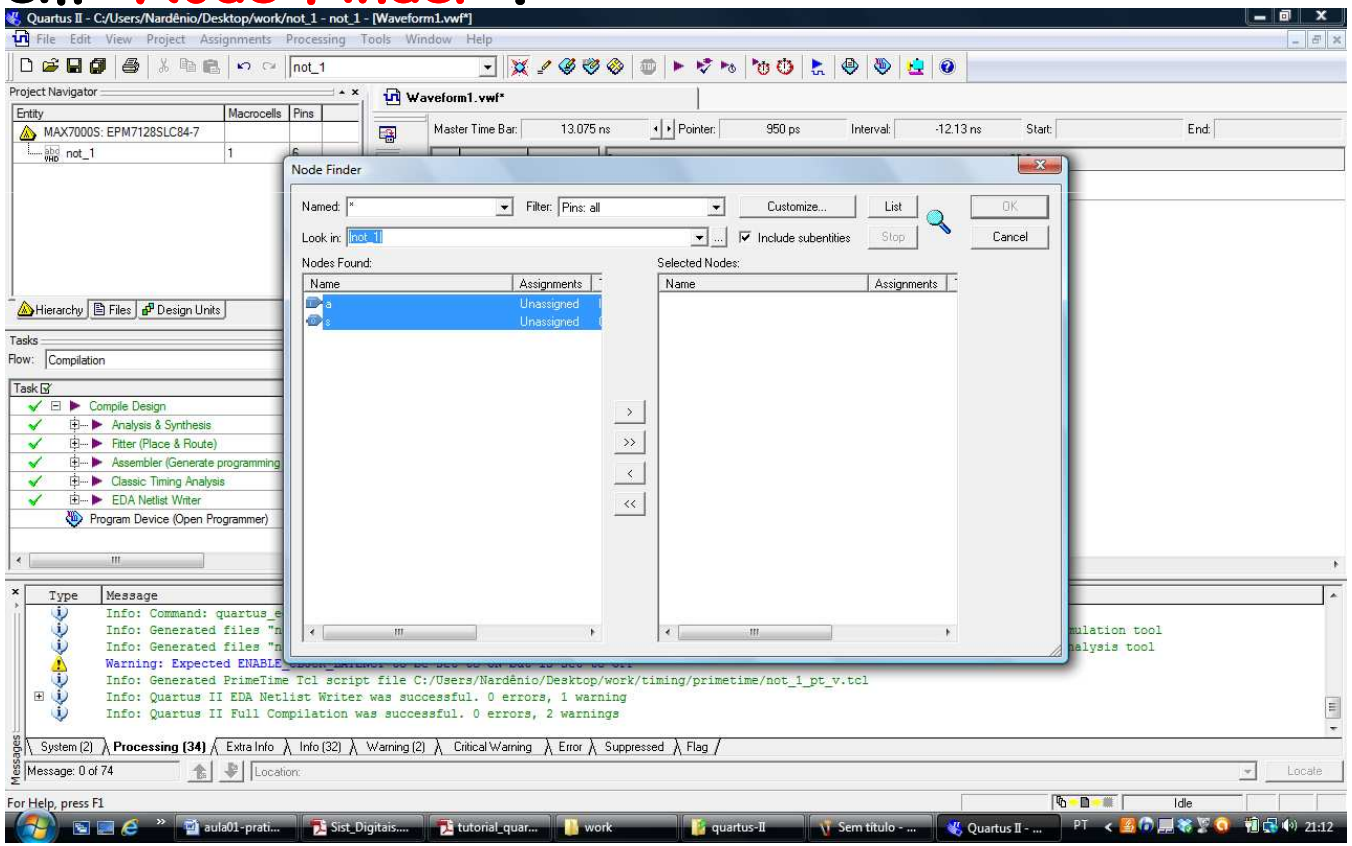
# Software Quartus II

18. Ajuste o tempo de simulação: "Edit > End Time" e coloque 80 ns para simulação. Clique "View > Fit in Window" para que todo o tempo de simulação fique visível na janela.



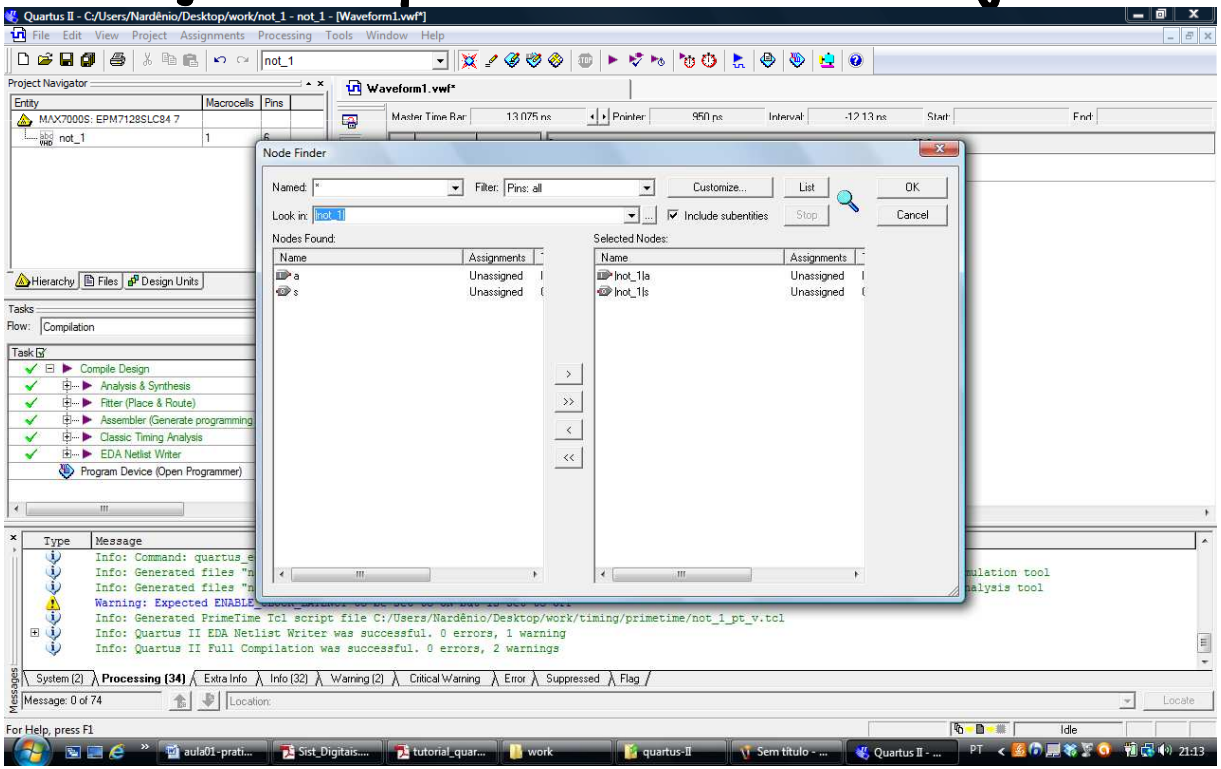
# Software Quartus II

19. Selecione os vetores de entrada e saída a serem incluídos na simulação. Para isto clique em "Edit > Insert > Node or Bus". Na janela que aparece, clique em "Node Finder".



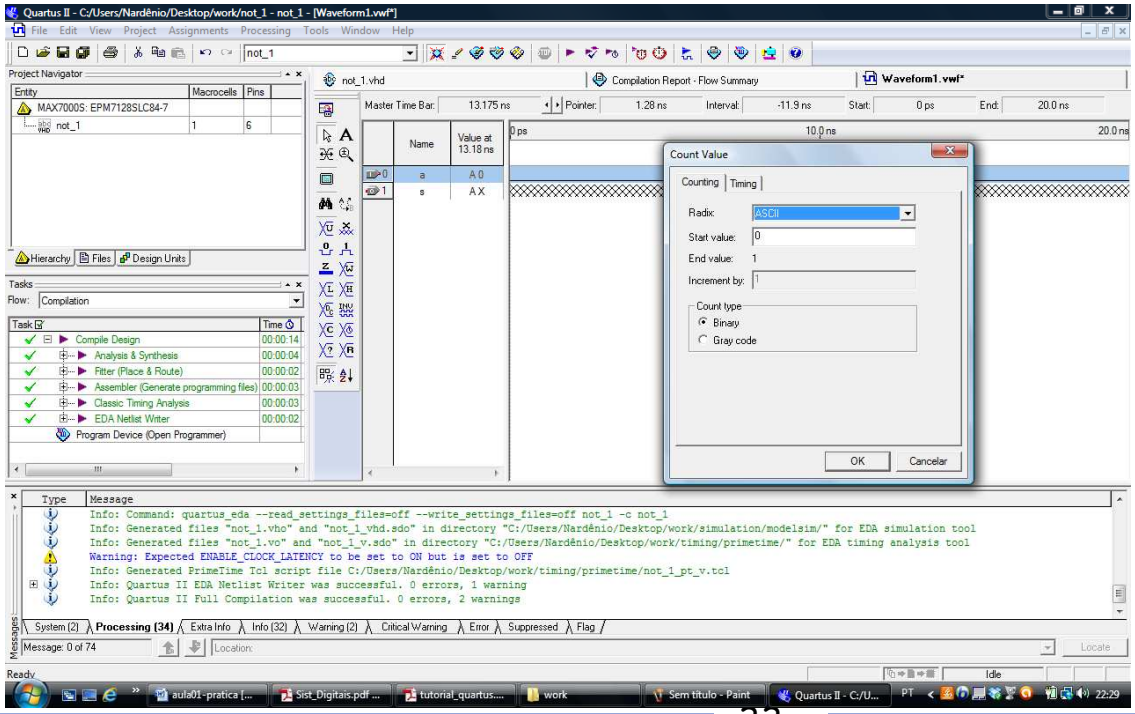
# Software Quartus II

20. Na próxima janela, selecione "Pins: All" e, em seguida, clique em "List" (a função *List* amostra os vetores de entrada e saída). Em seguida, utilizando o botão ">>", transferir as entradas e saídas para a ferramenta de simulação. Clique em Ok nas duas janelas subsequentes.



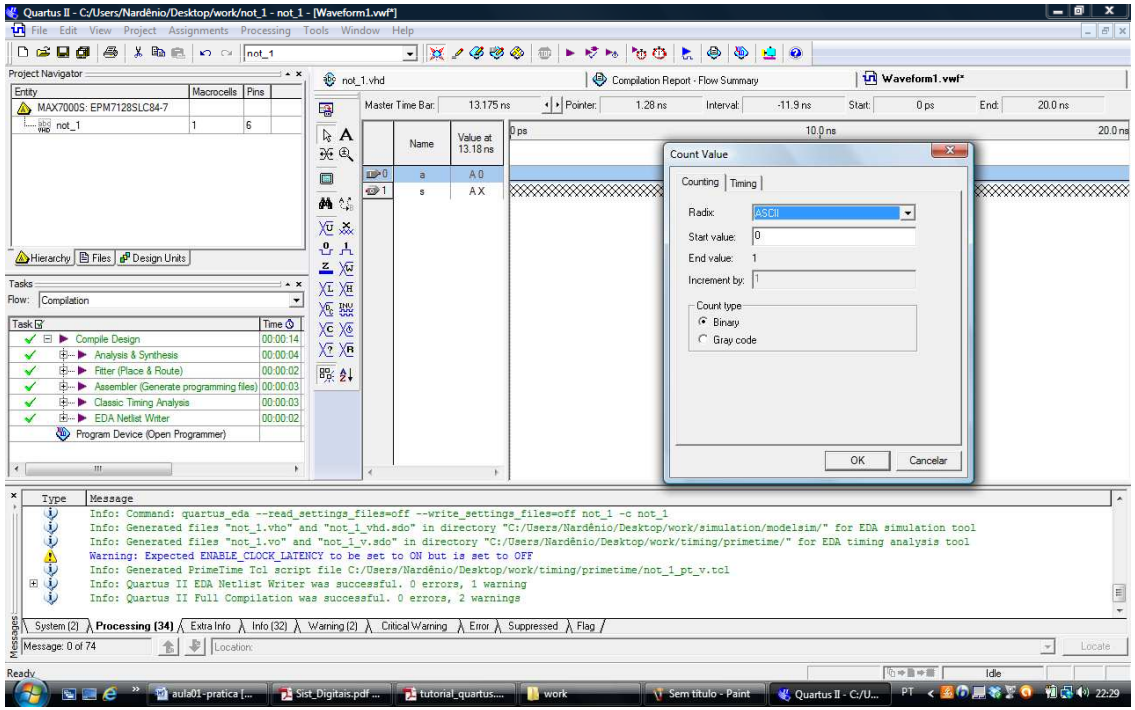
# Software Quartus II

21. Insira as formas de onda de entrada para testar todas as possibilidades para a(s) entrada(s) do projeto. Marque toda(s) a(s) entrada(s) do projeto, clique com o botão direito e selecione "**Grouping > Group**". Insira um nome para o grupo de entradas (por exemplo, "**inputs**" ou "**entradas**").



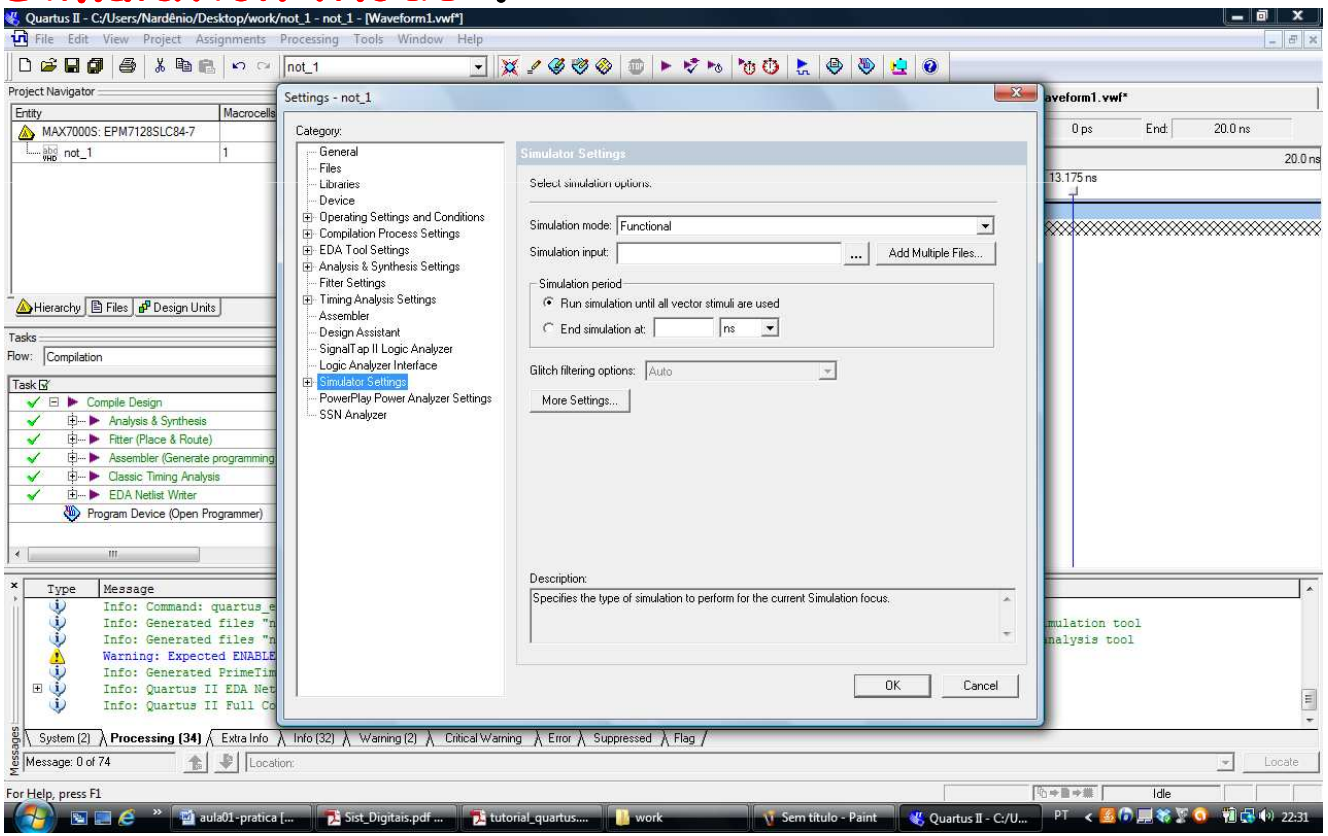
# Software Quartus II

22. Clique com o botão direito sobre a(s) entrada(s) e selecione "Value > Count Value". Verifique que o campo Start Value tenha o valor [0] e o End Value, [7] (na realidade, pode-se ver que os bits de entrada como três entradas de 1 *bit*, que pode assumir, portanto, valores de 000 a 111).



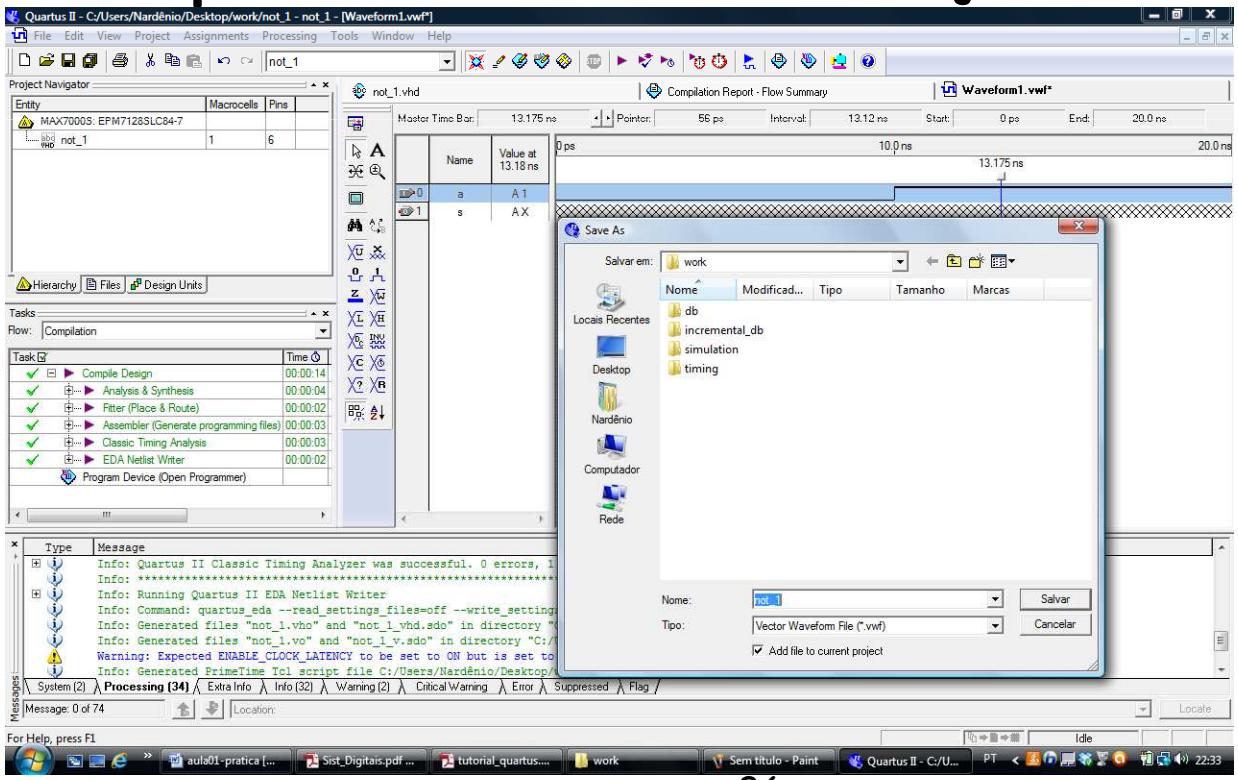
# Software Quartus II

23. O passo seguinte é a simulação do circuito projetado. Clique em "Assignments > Settings", selecione "Simulator Settings" e escolha "Functional em Simulation Mode".



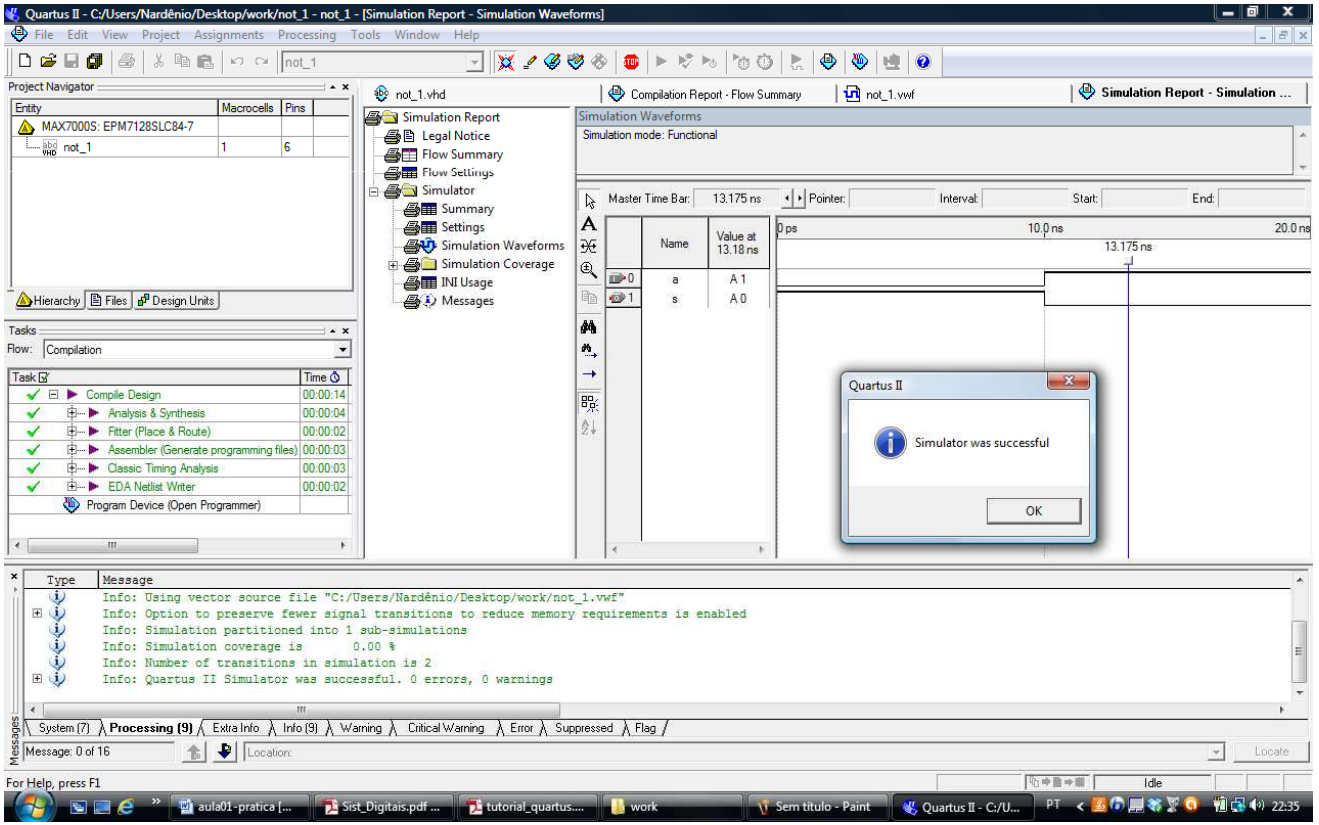
# Software Quartus II

24. Clique em "Processing > Generate Functional Simulation Netlist". Antes de executar a simulação é necessário salvar o arquivo que deve conter o mesmo nome dado ao código em VHDL. Neste caso, "mux2_est1.vwf". Esses passos definem uma simulação funcional.



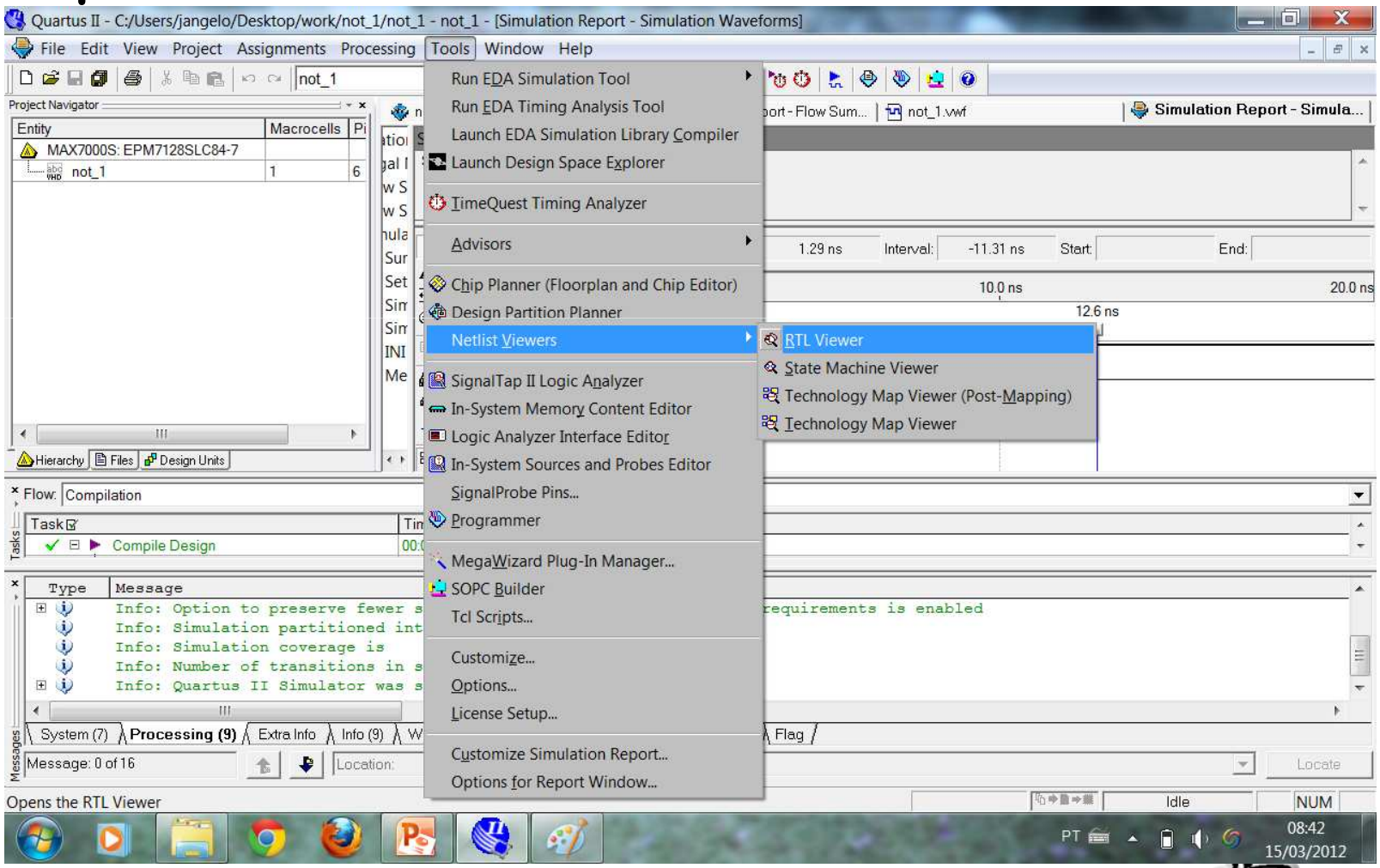
# Software Quartus II

25. Clique em "Processing > Start Simulation". Verifique o valor da saída para cada entrada e veja que o circuito sintetizado a partir do código em VHDL de fato implementa a função desejada.



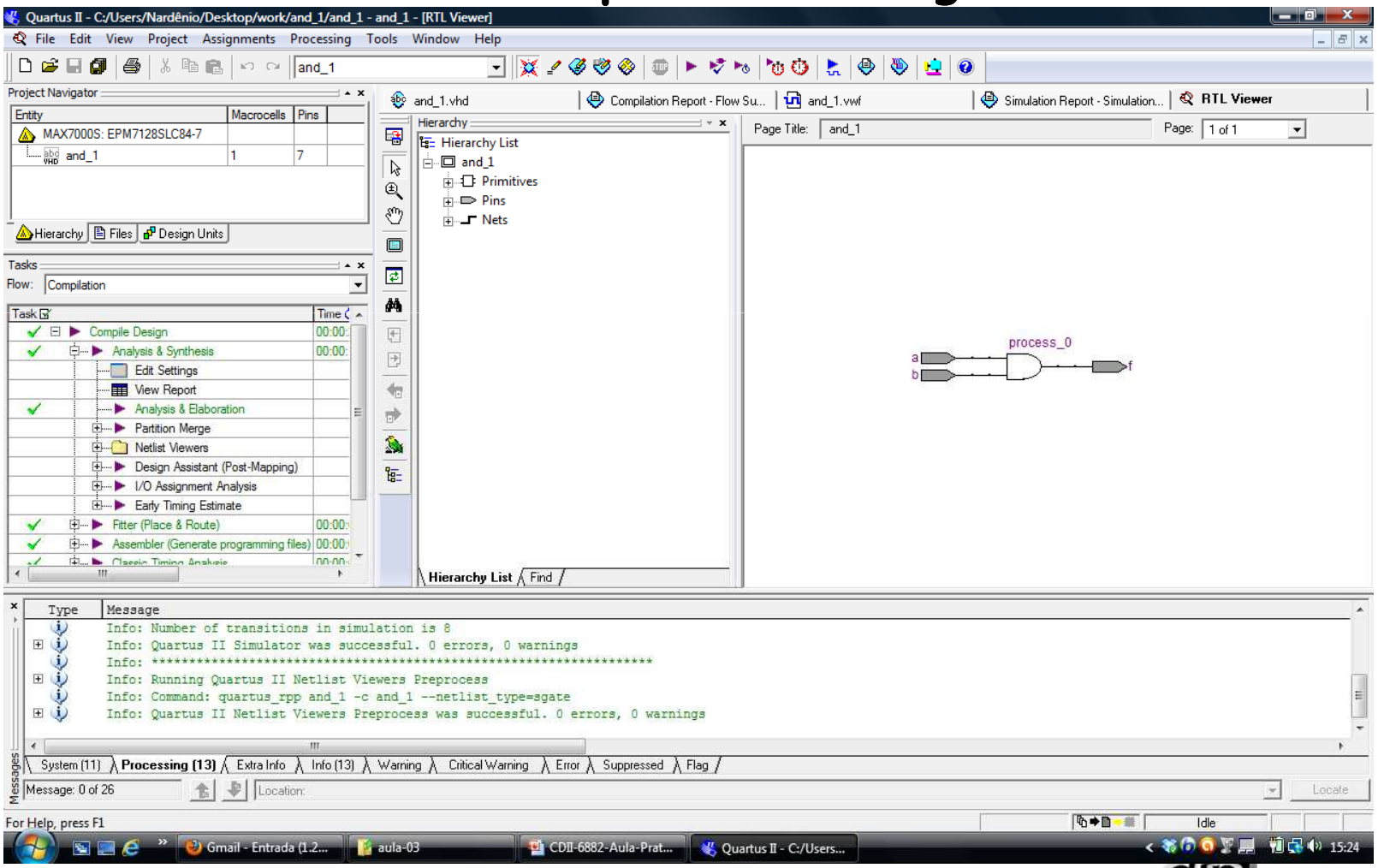
# Software Quartus II

26. Clique em "Tools > Netlist Viewers > RTL Viewer".



# Software Quartus II

## 27. O circuito sintetizado a partir do código em VHDL.



# Aula de Hoje

Repita os procedimentos para as implementações das demais portas lógicas

# HDL – Linguagem de Descrição de Hardware

## Modelagem comportamental em VHDL usando IF THEN ELSE

```
ARCHITECTURE comportamental OF mux2_com IS
BEGIN
    PROCESS (a, b, s)
    BEGIN
        IF s='0' THEN
            f<=a;
        ELSE
            f<=b;
        END IF;
     END PROCESS;
END comportamental;
```

# HDL – Linguagem de Descrição de Hardware

## Modelagem comportamental em VHDL usando WHEN ELSE

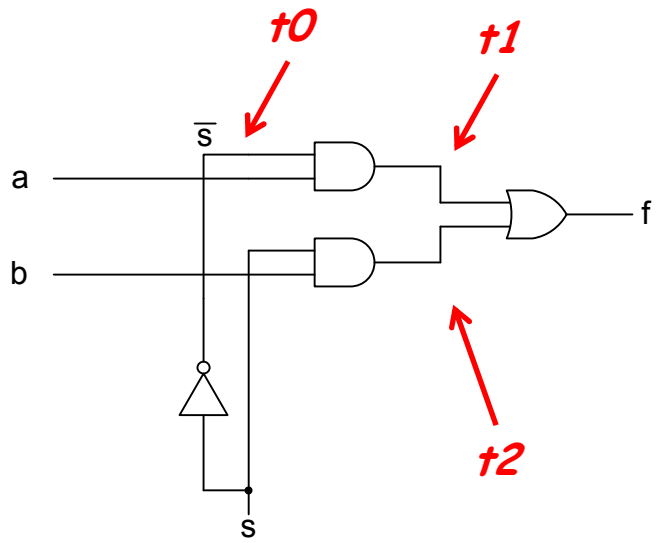
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY mux2_com1 IS
    PORT (a, b : IN BIT;
          s    : IN BIT;
          f    : OUT BIT);
END mux2_com1;
ARCHITECTURE comportamental  OF mux2_com1 IS
BEGIN
        f <= a WHEN s = '0' ELSE b;
END comportamental;
```

# HDL – Linguagem de Descrição de Hardware

## Modelagem estrutural em VHDL

- **Solução 01:**

```
ARCHITECTURE estrutural OF mux2_est IS
SIGNAL t0, t1, t2 : BIT;
BEGIN
        t0 <= NOT s;
        t1 <= t0 AND a;
        t2 <= s AND b;
         f <= t1 OR t2;
END estrutural;
```



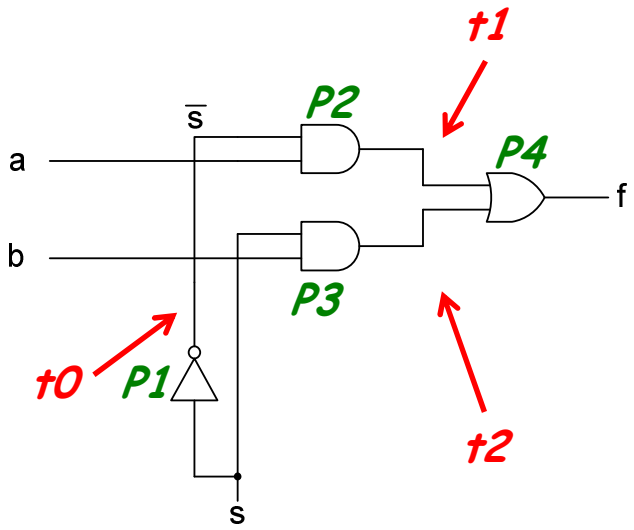

- A descrição estrutural especifica separadamente a função de cada porta no multiplexador.
- Essa descrição é baseada na expressão booleana do circuito.

# HDL – Linguagem de Descrição de Hardware

## Modelagem estrutural em VHDL usando componentes

- **Solução 02:**

```
ENTITY not1 IS
    PORT (x : IN BIT;
          z : OUT BIT);
END not1;
ARCHITECTURE logica OF not1 IS
BEGIN
        z <= NOT x;
END logica;
```

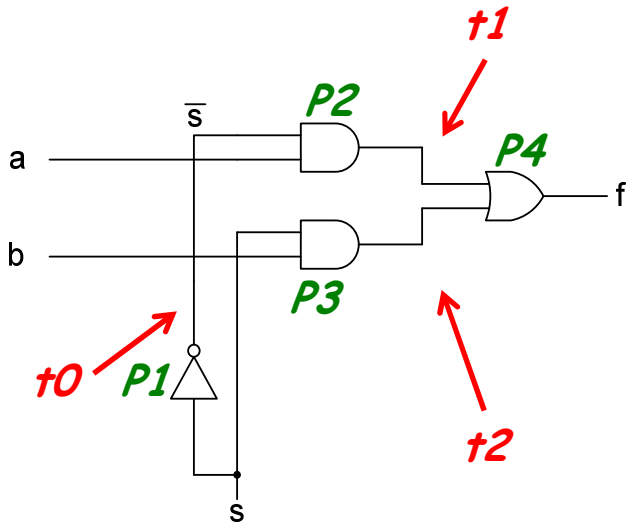



- A descrição estrutural especifica separadamente a função de cada porta no multiplexador.
- Essa descrição é baseada na expressão booleana do circuito.

# HDL – Linguagem de Descrição de Hardware

## Modelagem estrutural em VHDL usando componentes

```
ENTITY and2 IS
      PORT (x, y : IN BIT;
                   z : OUT BIT);
END and2;
ARCHITECTURE logica OF and2 IS
BEGIN
            z <= x AND y;
END logica;
```

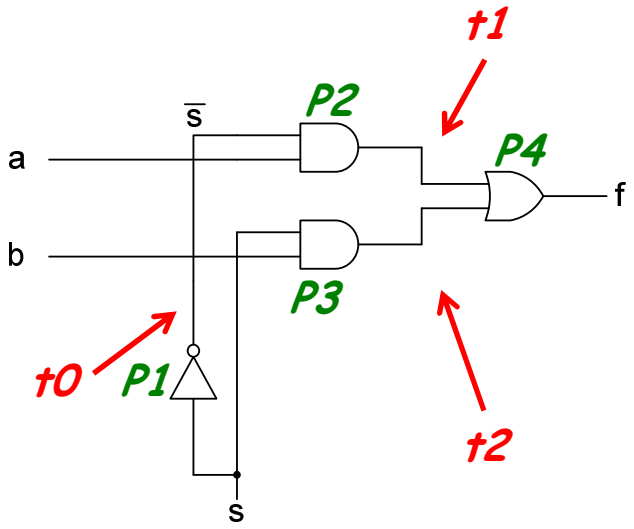



- A descrição estrutural especifica separadamente a função de cada porta no multiplexador.
- Essa descrição é baseada na expressão booleana do circuito.

# HDL – Linguagem de Descrição de Hardware

## Modelagem estrutural em VHDL usando componentes

```
ENTITY or2 IS
        PORT (x, y : IN BIT;
                     z : OUT BIT);
END or2;
ARCHITECTURE logica OF or2 IS
BEGIN
            z <= x OR y;
END logica;
```




- A descrição estrutural especifica separadamente a função de cada porta no multiplexador.
- Essa descrição é baseada na expressão booleana do circuito.

# HDL – Linguagem de Descrição de Hardware

## Modelagem estrutural em VHDL usando componentes

```
ENTITY mux2_est1 IS
    PORT (a, b : IN BIT;
          s : IN BIT;
          f : OUT BIT);
END mux2_est1;
ARCHITECTURE estrutural OF mux2_est1 IS
COMPONENT not1
    PORT (x : IN BIT;
          z : OUT BIT);
END component;
COMPONENT and2
    PORT (x, y : IN BIT;
          z : OUT BIT);
END component;                    -- continuacao
```

```
COMPONENT or2      -- continuacao
    PORT (x, y : IN BIT;
          z : OUT BIT);
END component;

SIGNAL t0, t1, t2 : BIT;
BEGIN
    P1: not1 PORT MAP (s, t0);
    P2: and2 PORT MAP (t0, a, t1);
    P3: and2 PORT MAP (s, b, t2);
    P4:  or2 PORT MAP (t1, t2, f);
END estrutural;
```