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

usando os comandos:

CASE WHEN

WITH SELECT WHEN

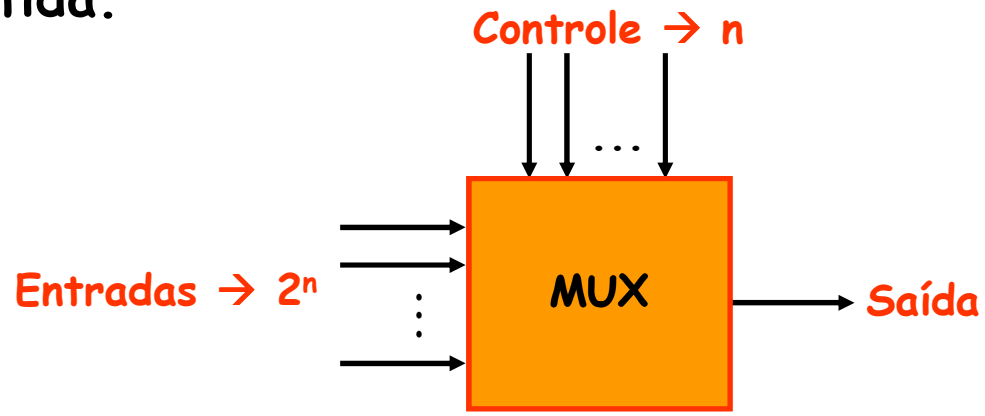
WHEN ELSE

HDL – Linguagem de Descrição de Hardware

Multiplexador

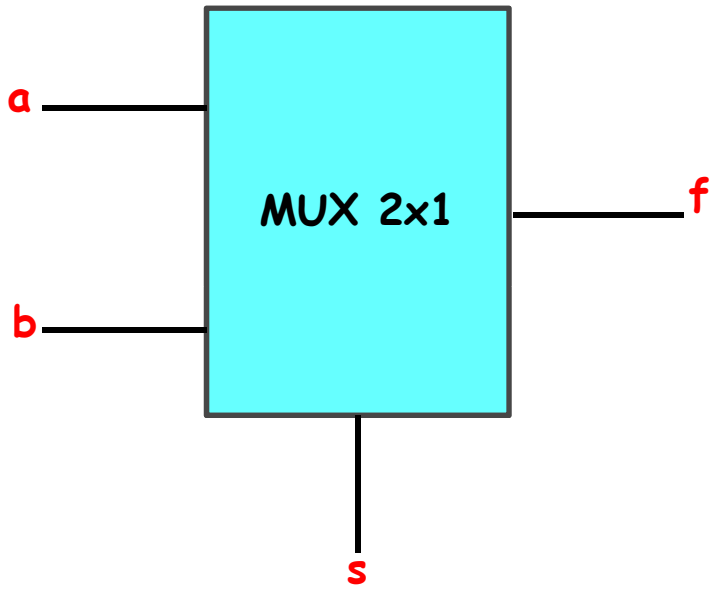
Multiplexador ou Seletor de Dados: É um circuito lógico que tem diversas entradas e apenas uma saída. MUX seleciona uma única entrada para transmitir para a saída.

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



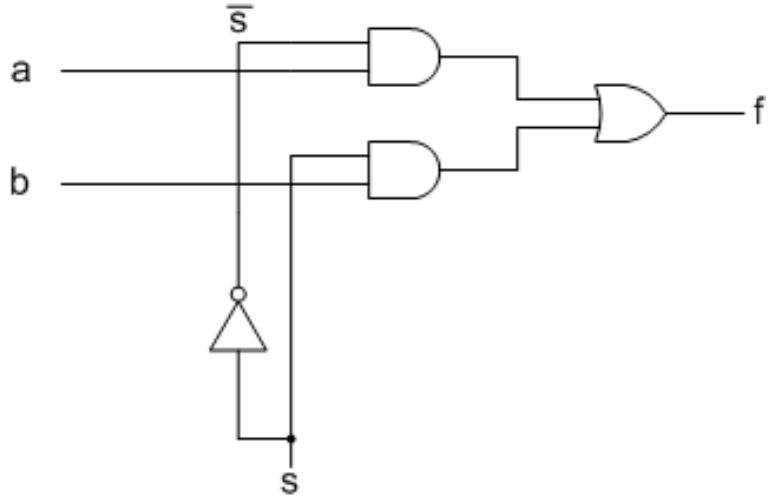
HDL – Linguagem de Descrição de Hardware

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

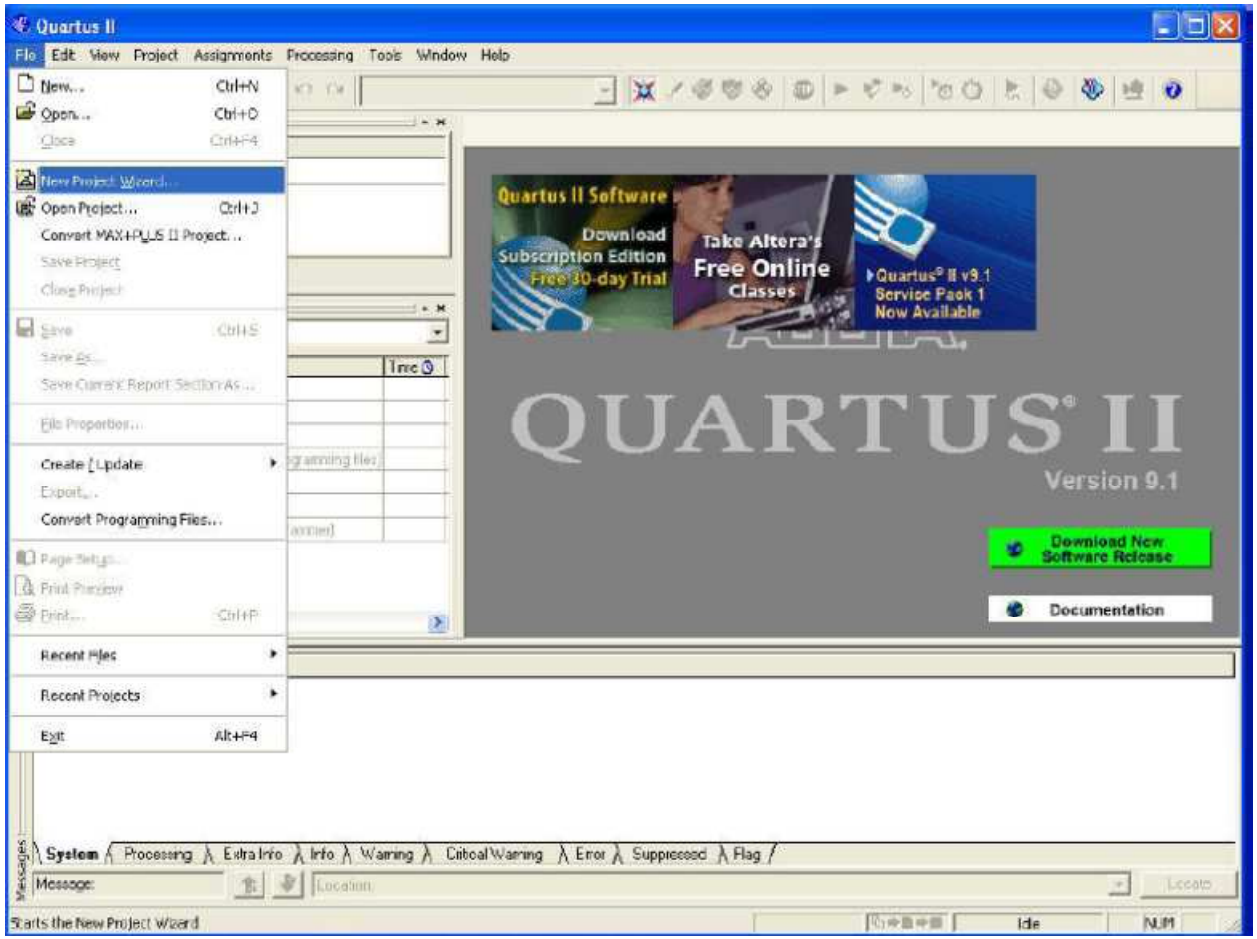
LIBRARY IEEE; **USE IEEE.STD_LOGIC_1164.all;** **USE IEEE.STD_LOGIC_UNSIGNED.all;**	LIBRARY (PACOTES)
ENTITY exemplo **IS** **PORT** (<descrição dos pinos de I/O>); **END** exemplo;	ENTITY (PINOS DE I/O)
ARCHITECTURE teste **OF** exemplo **IS** **BEGIN** ... **END** teste;	ARCHITECTURE (ARQUITETURA)

Software Quartus II

1. Crie diretório ou pasta "work" na área de trabalho.
2. Crie os seguintes subdiretórios dentro do diretório "work":
   a) "mux_cc_case"
   b) "mux_cc_with"
   c) "cod_cc_when" → Codificador Decimal
3. Inicialize o Software Quartus II

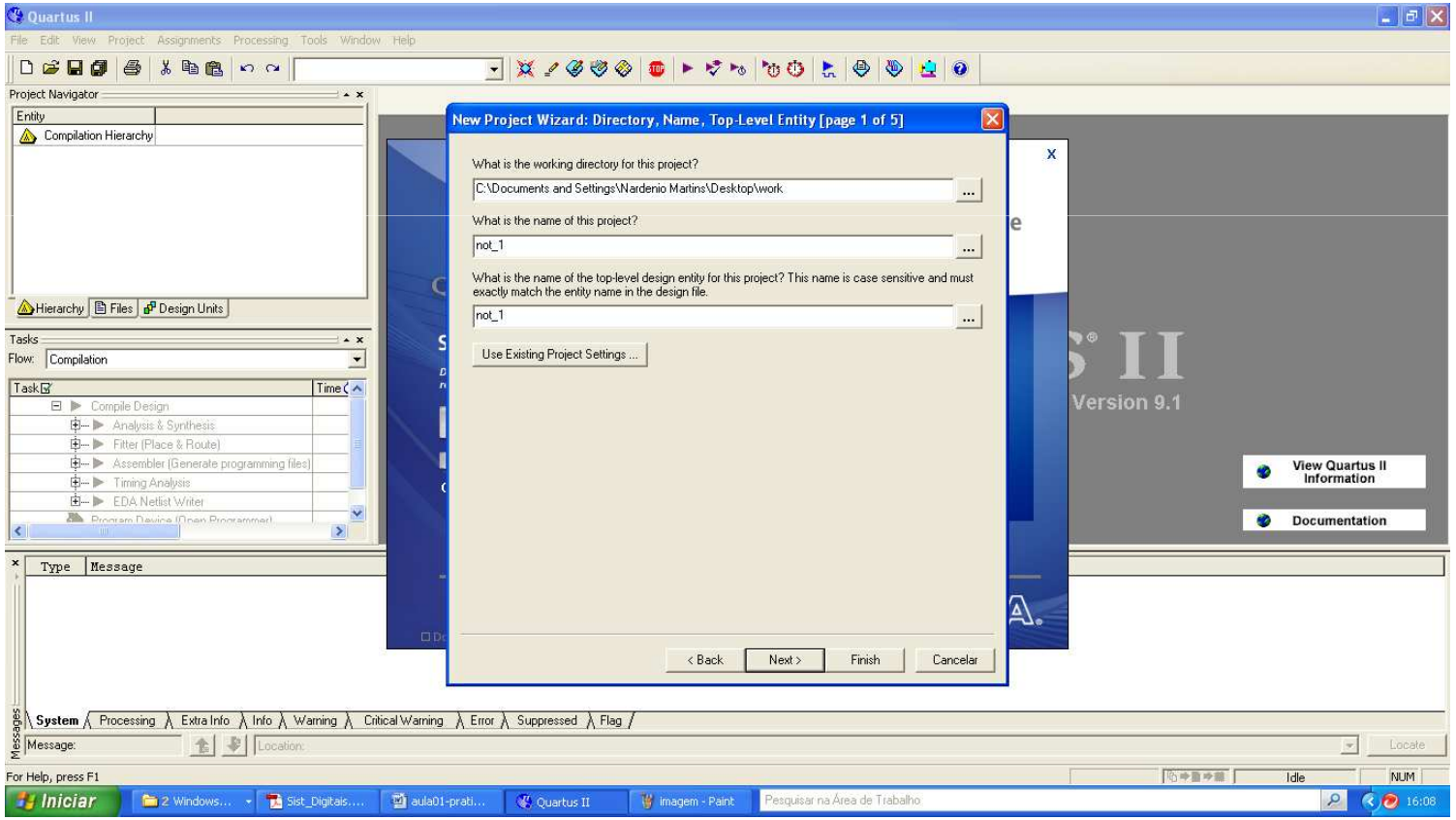
Software Quartus II

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



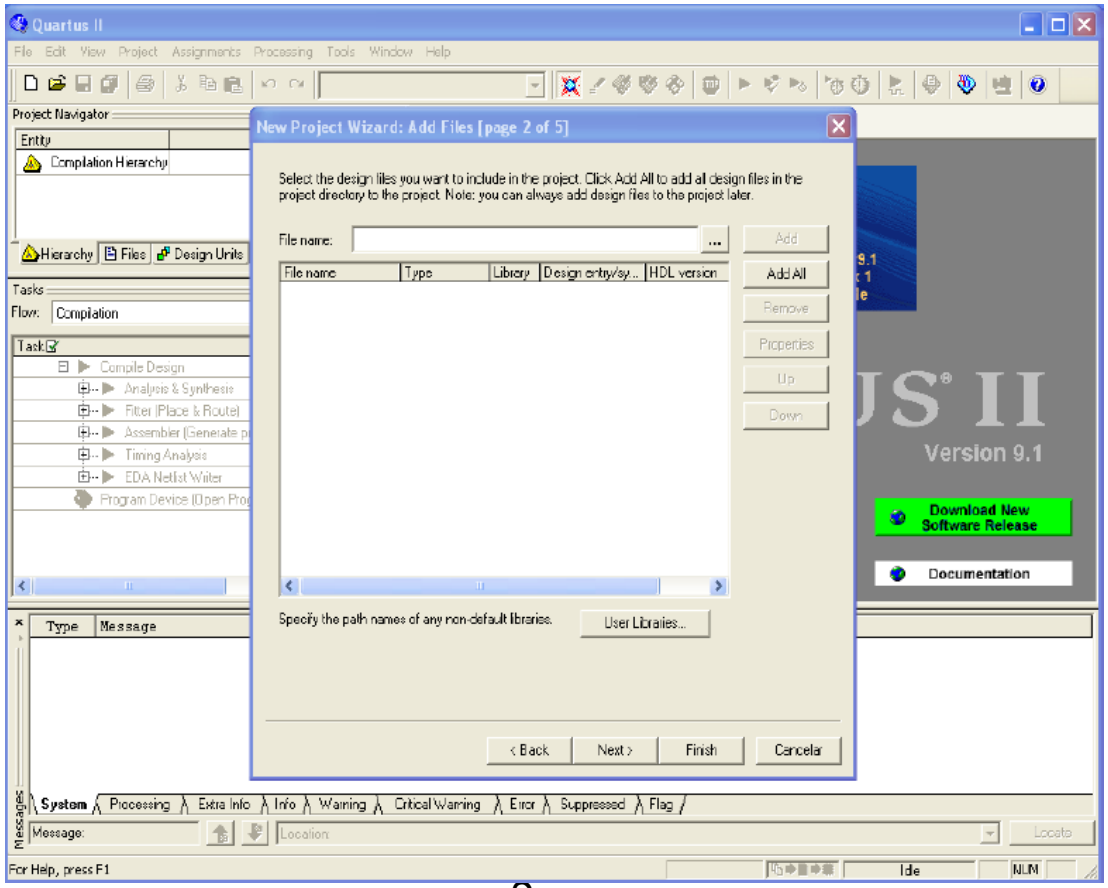
Software Quartus II

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 → "mux_cc_case".



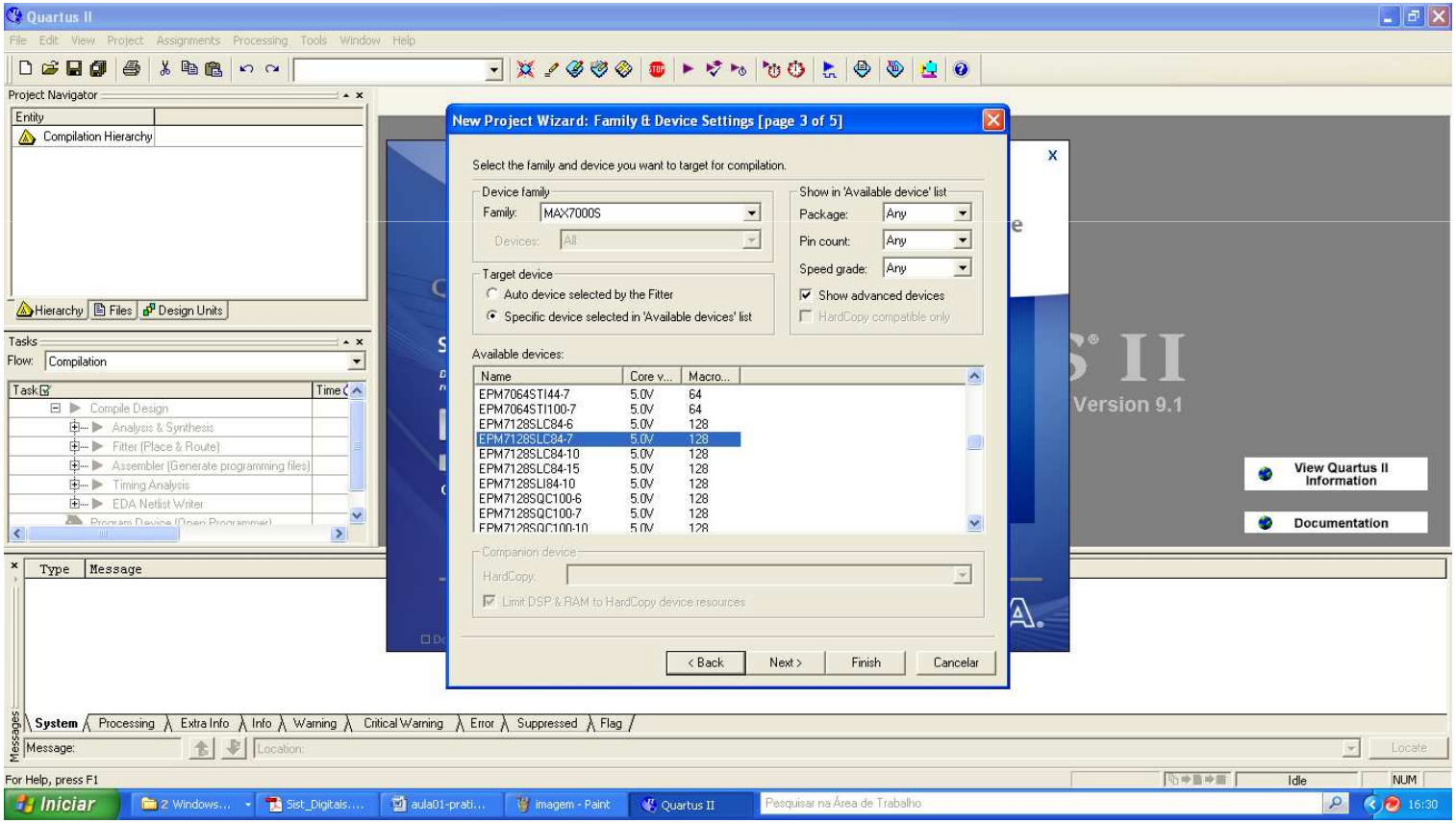
Software Quartus II

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



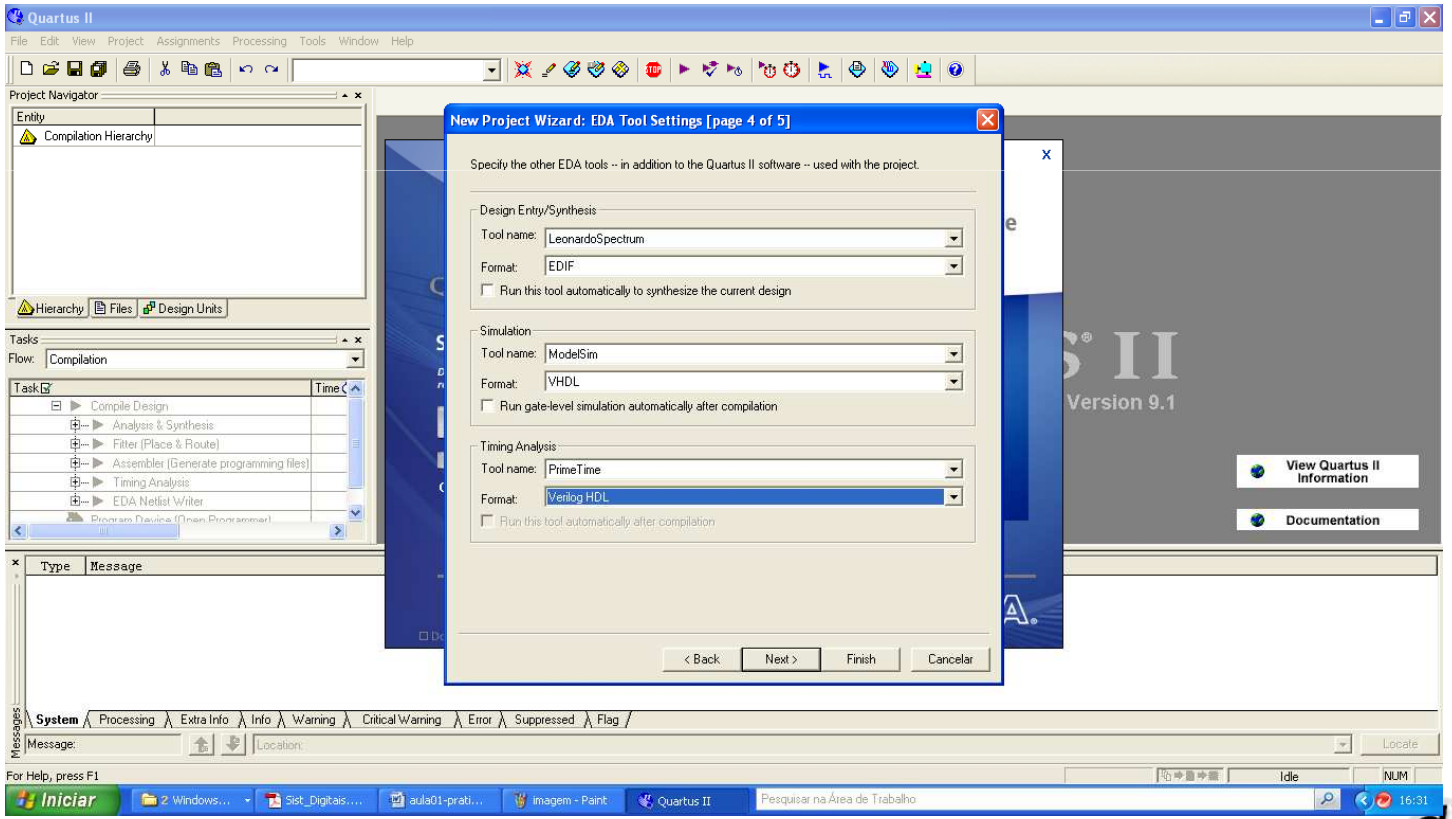
Software Quartus II

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



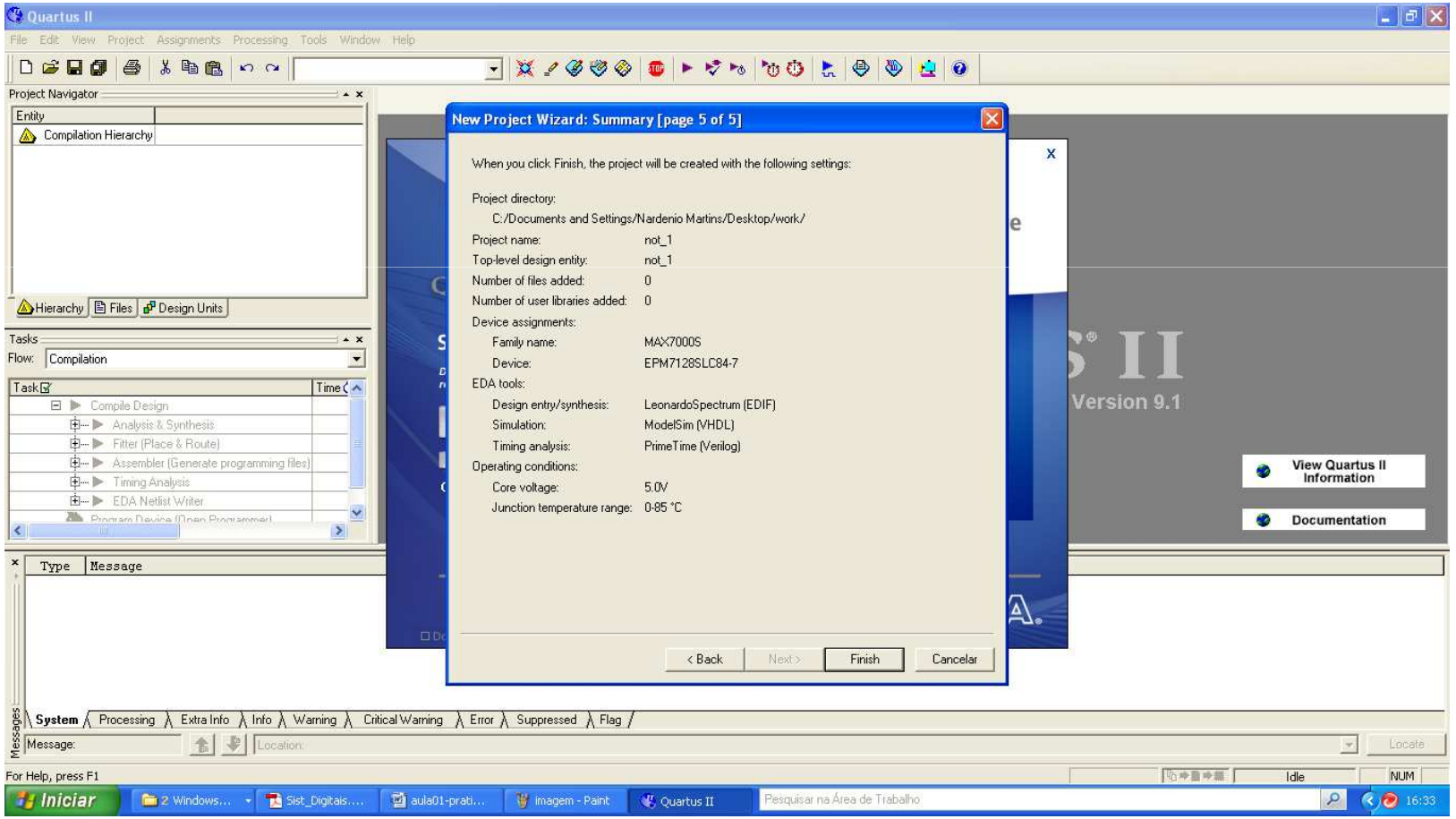
Software Quartus II

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.



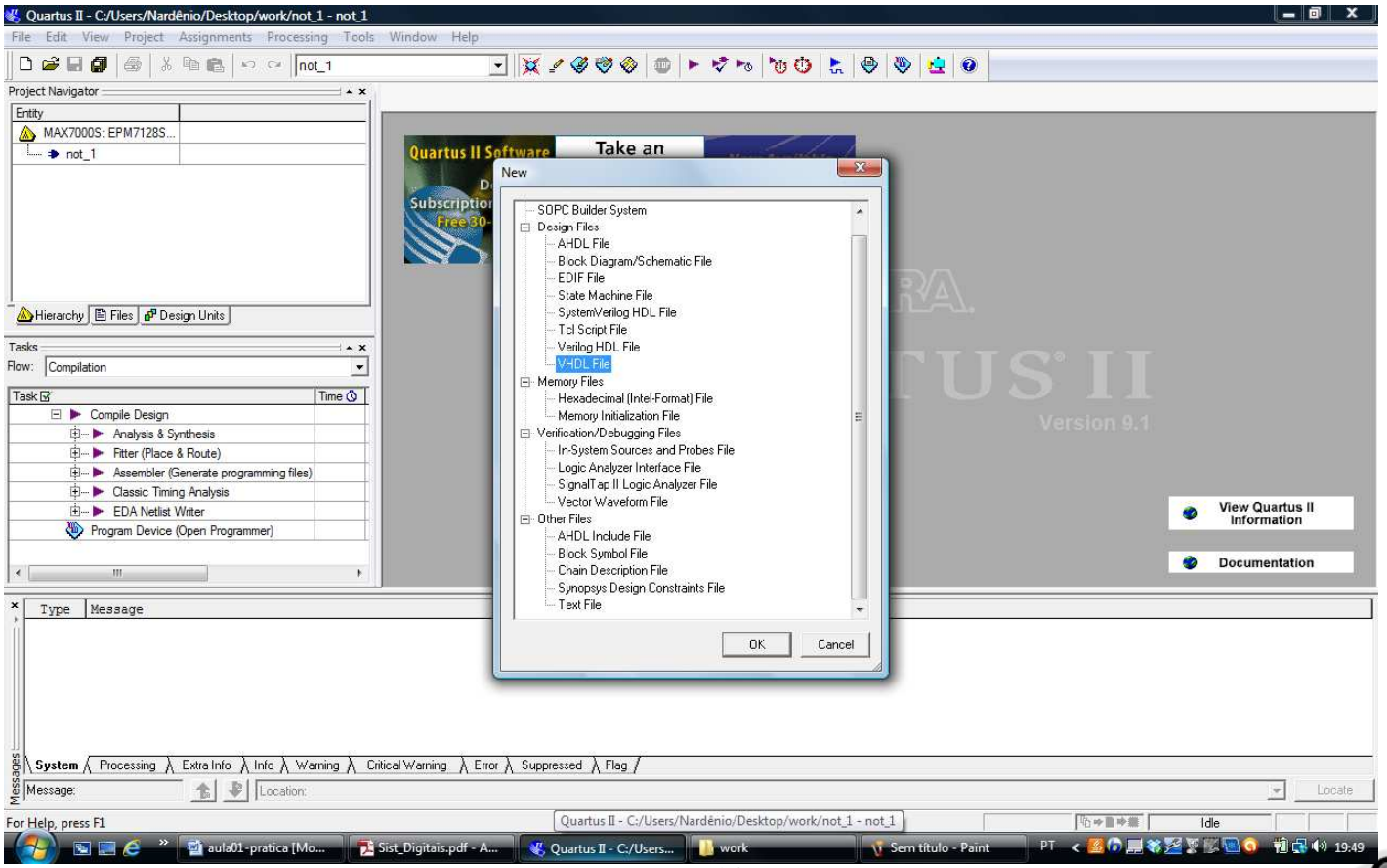
Software Quartus II

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".



HDL – Linguagem de Descrição de Hardware

CASE WHEN

- É um comando sequencial com uso dentro de procedimentos, funções e processos.
- Permite a definição de várias condições em um componente.
- Neste comando, as comparações sempre são feitas em torno de um único objeto ou expressão, e será o valor desse objeto ou determinada condição que indicará quais comandos serão executados.
- Sintaxe:

```
CASE expressao_de_escolha IS                          -- expressao_de_escolha =
   WHEN condicao_1                  => comando_a;             -- condicao_1
   WHEN condicao_2                  => comando_b; comando_c;-- condicao_2
   WHEN condicao_3 | condicao_4     => comando_d;             -- condicao_3 ou condicao_4
   WHEN condicao_5 TO condicao_9    => comando_d;             -- condicao_5 ate condicao_9
   WHEN OTHERS                      => comando_e; comando_f;-- condicoes restantes
END CASE;
```

- NOTA: O delimitador | equivale a uma operação OU entre as condições de escolha. As palavras reservadas TO e DOWNTO servem para delimitar uma faixa de condições. A palavra reservada OTHERS na última condição serve para agrupar as condições não-relacionadas na lista.

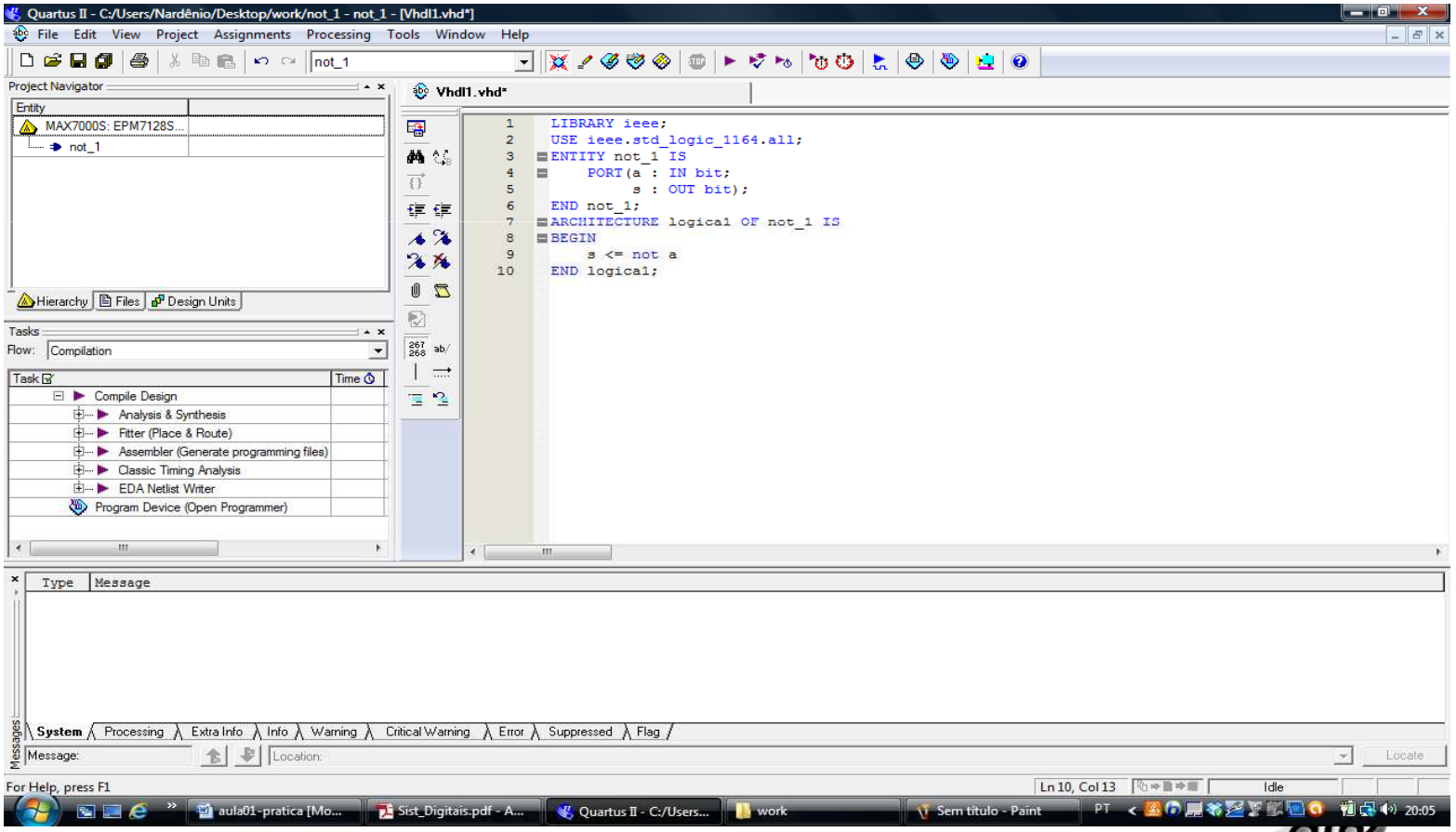
Software Quartus II

11.Escreva o código em VHDL.

```
LIBRARY ieee;          -- Multiplexador 2 x 1 usando comando case when
USE ieee.std_logic_1164.all;
ENTITY mux_cc_case IS
    PORT (a, b, s : IN BIT;
                f : OUT BIT);
END mux_cc_case;
ARCHITECTURE condicional OF mux_cc_case IS
BEGIN
          PROCESS (a, b, s)
          BEGIN
                    CASE s IS
                              WHEN '0' => f <= a;
                              WHEN '1' => f <= b;
                    END CASE;
          END PROCESS;
END condicional;
```

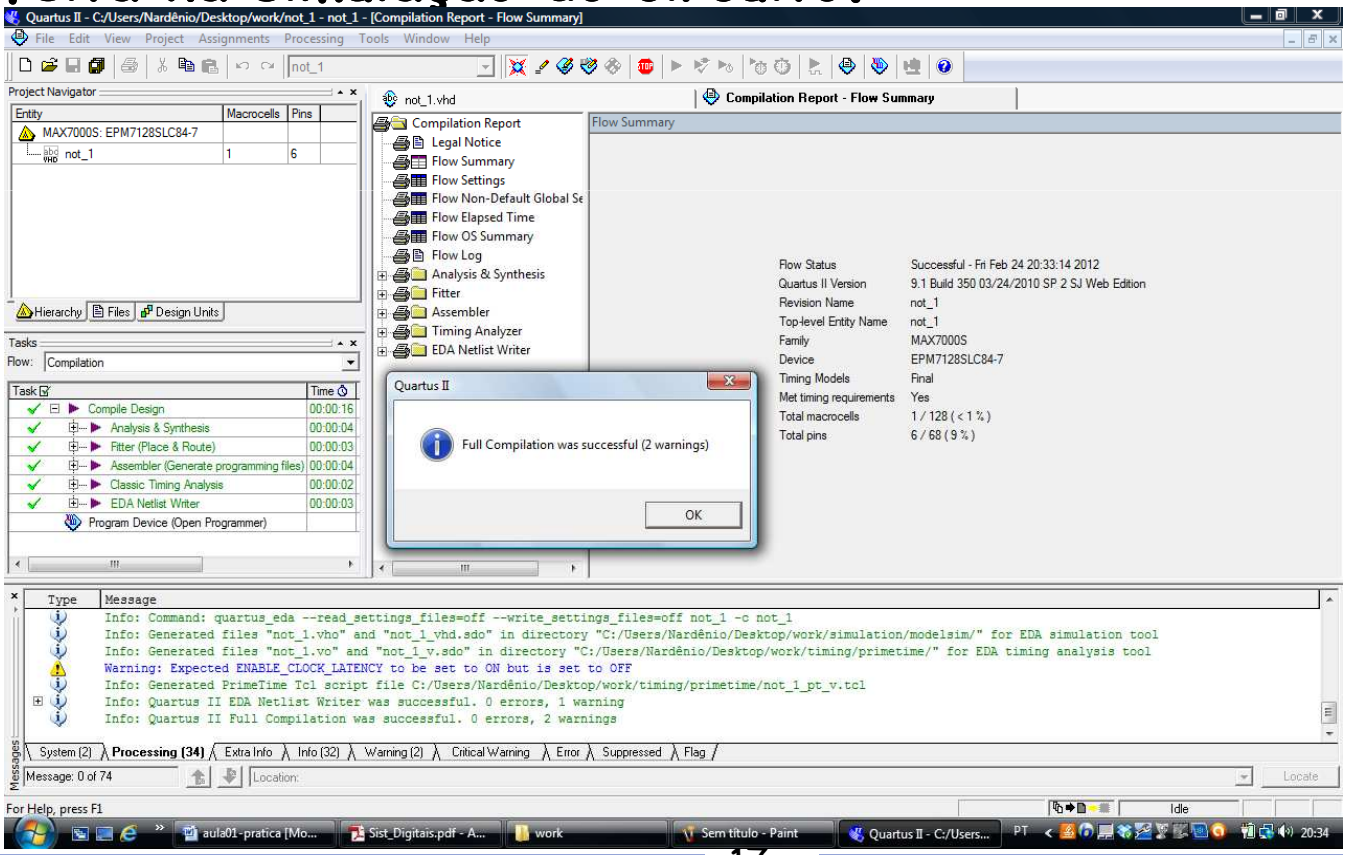
Software Quartus II

12. Salve o código em VHDL com extensão "mux_cc_case.vhd" na pasta ou subdiretório "mux_cc_case".



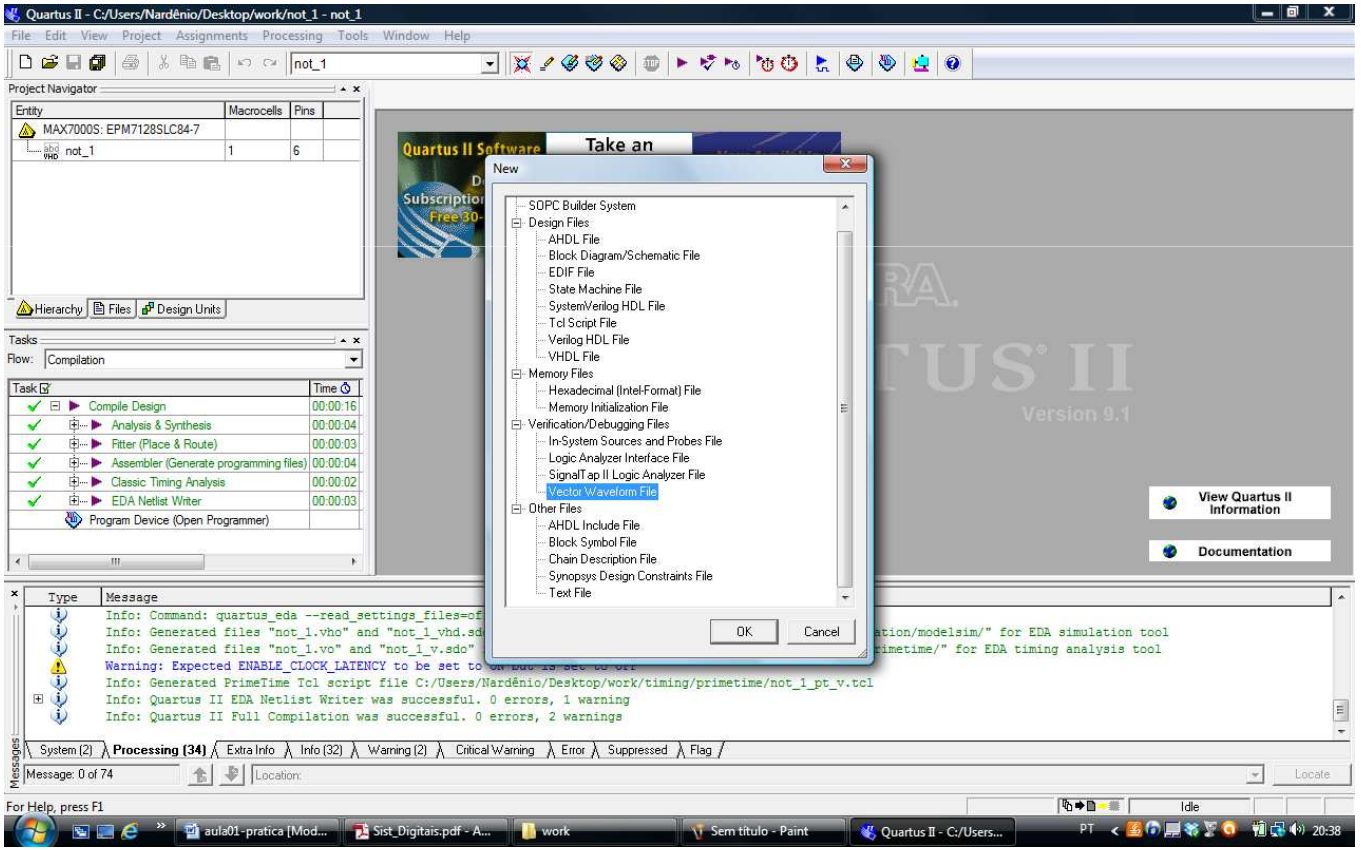
Software Quartus II

13. 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.



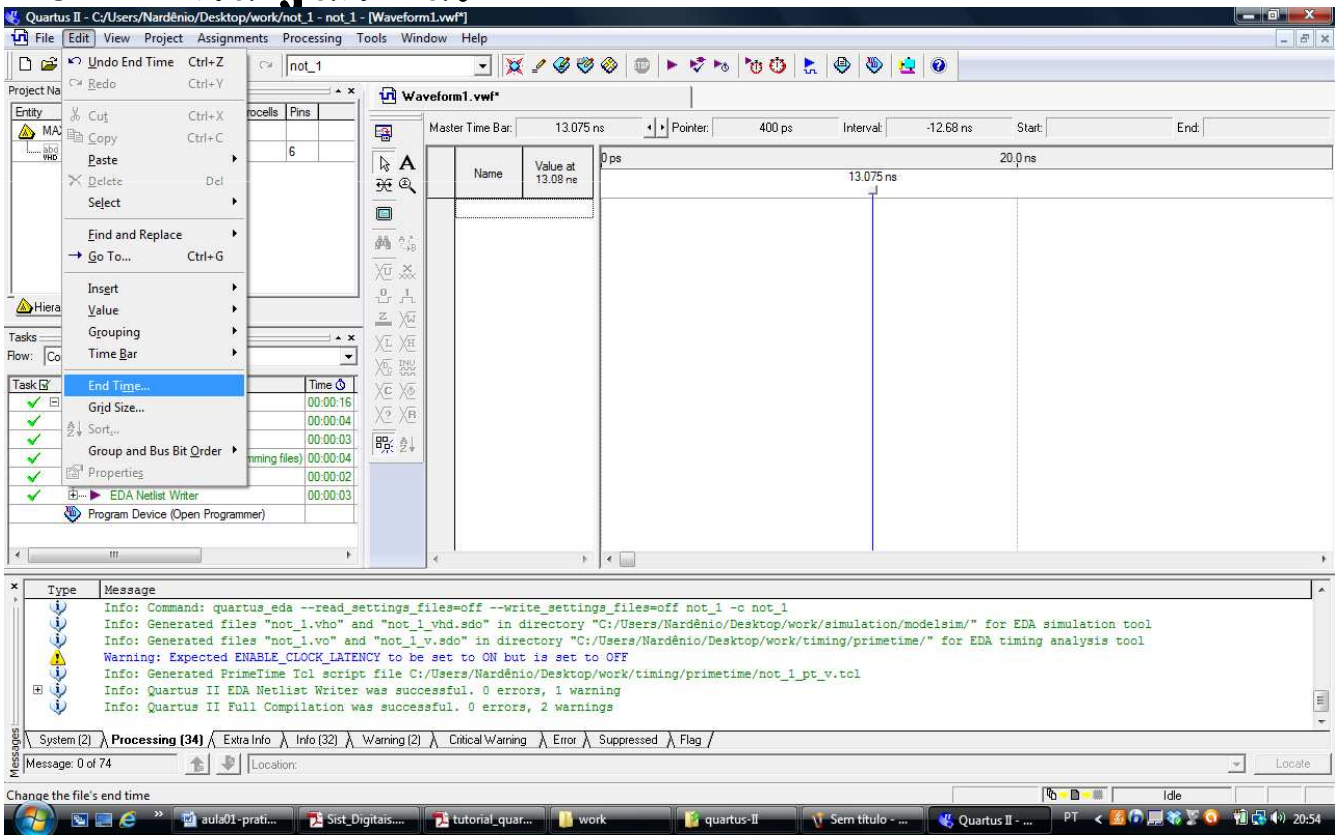
Software Quartus II

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



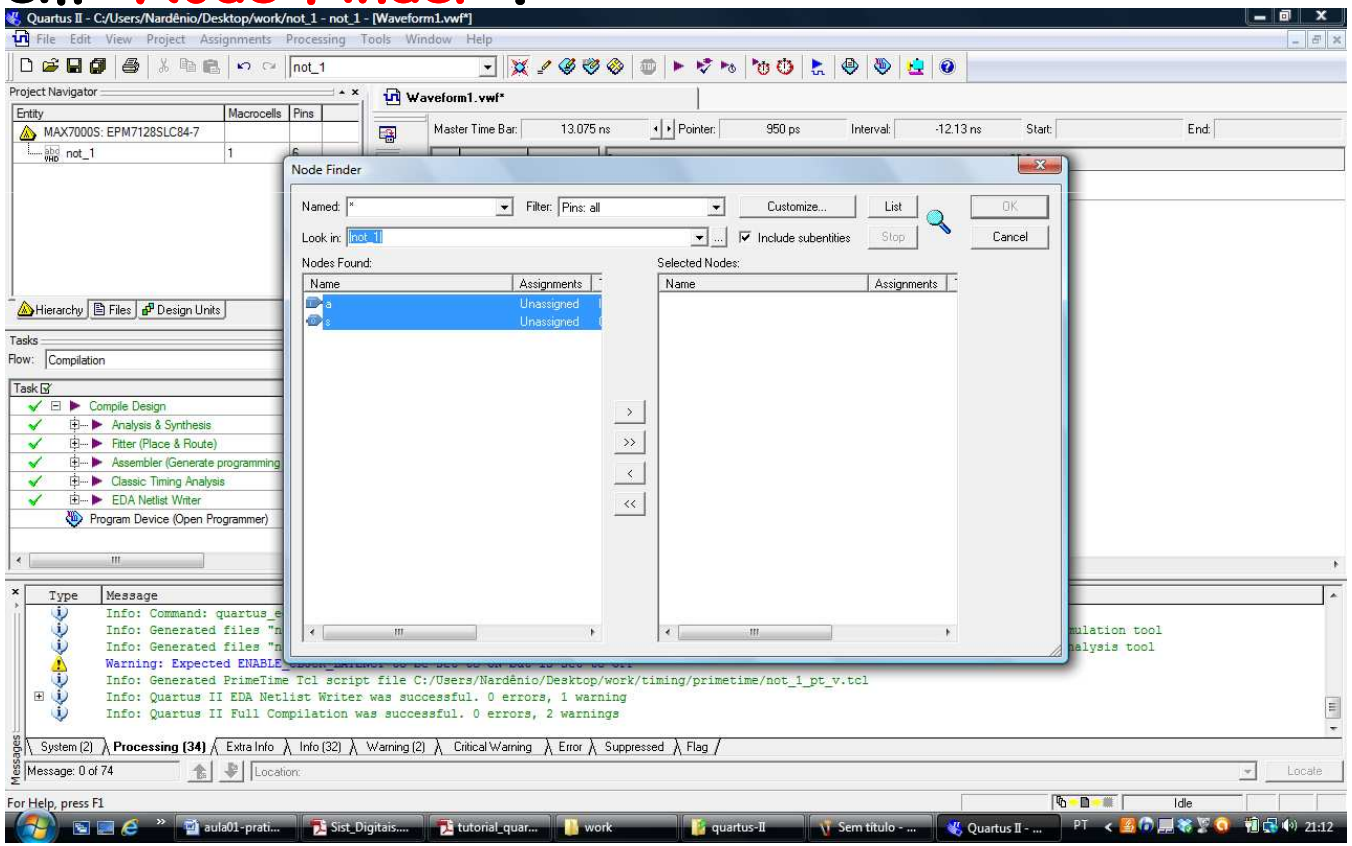
Software Quartus II

15. 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.



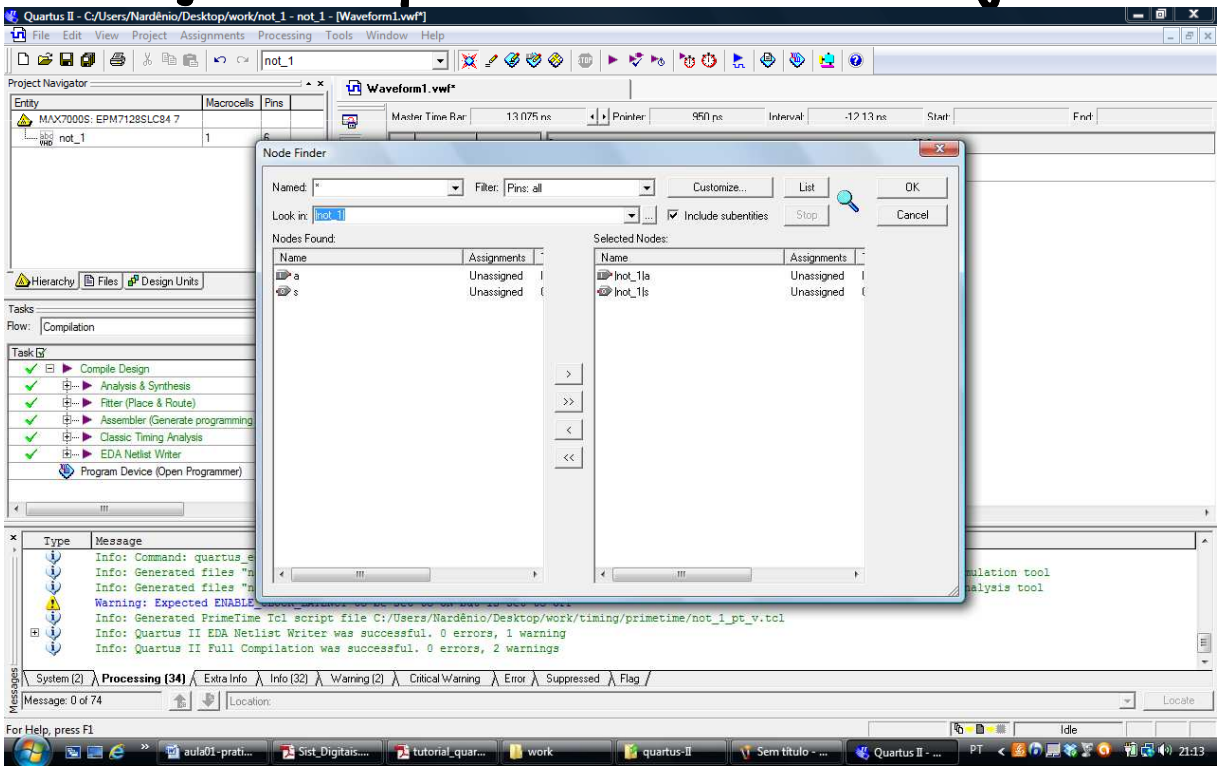
Software Quartus II

16. 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".



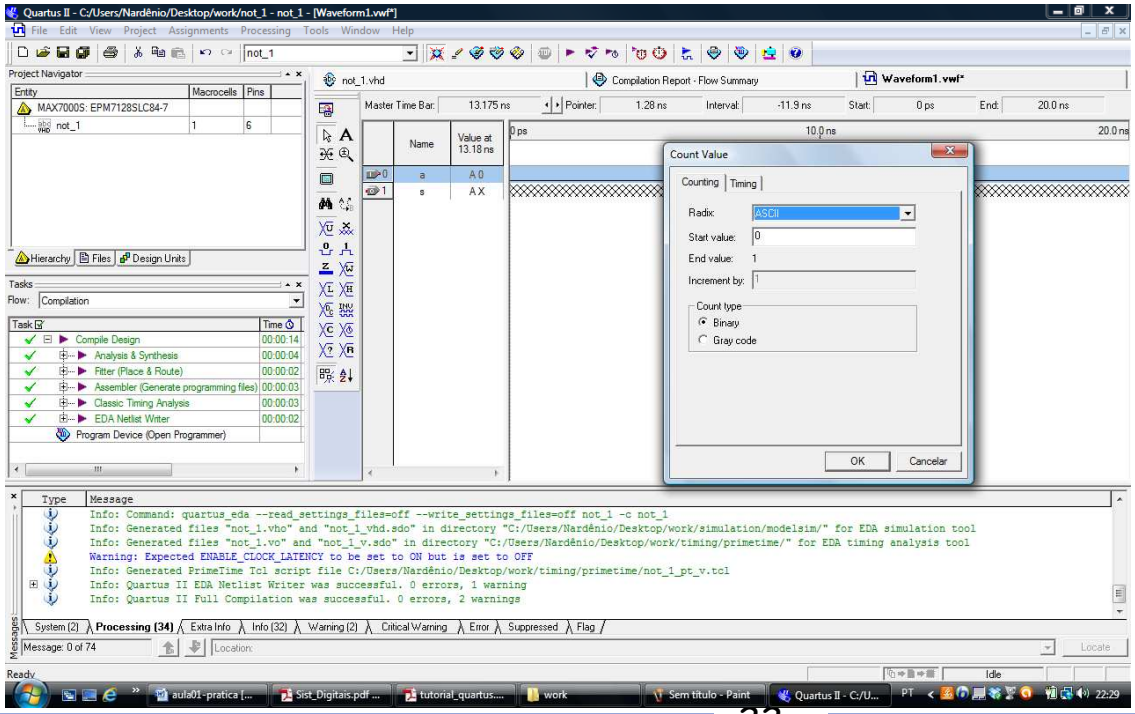
Software Quartus II

17. 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.



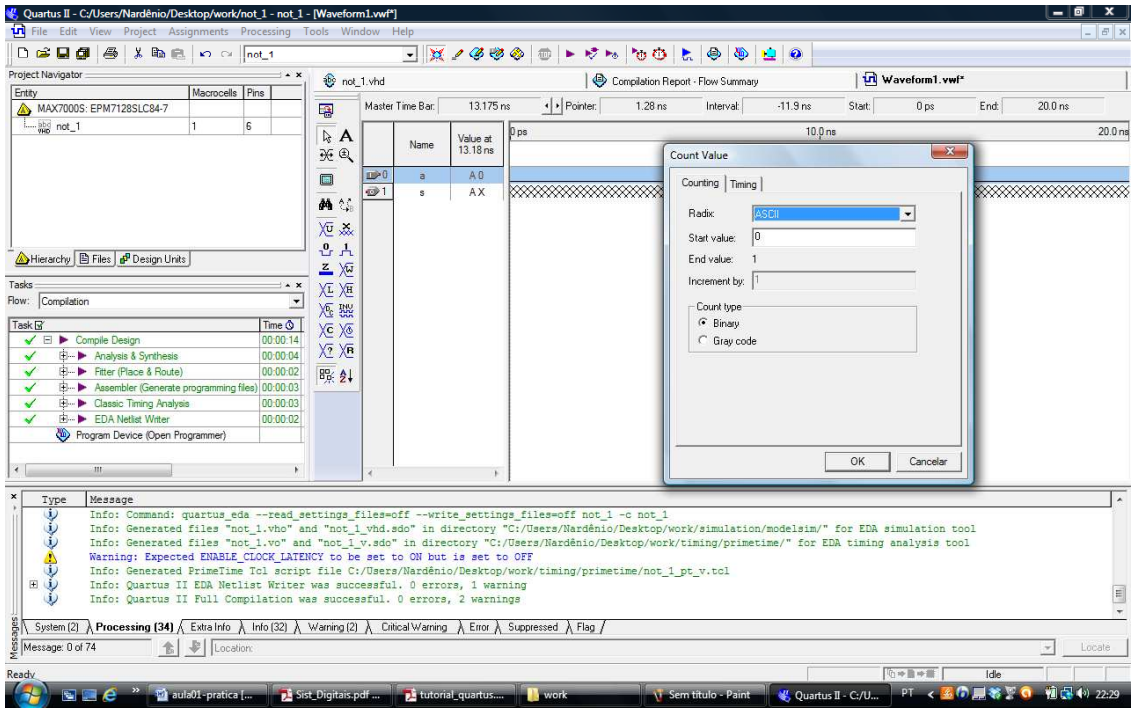
Software Quartus II

18. 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**").



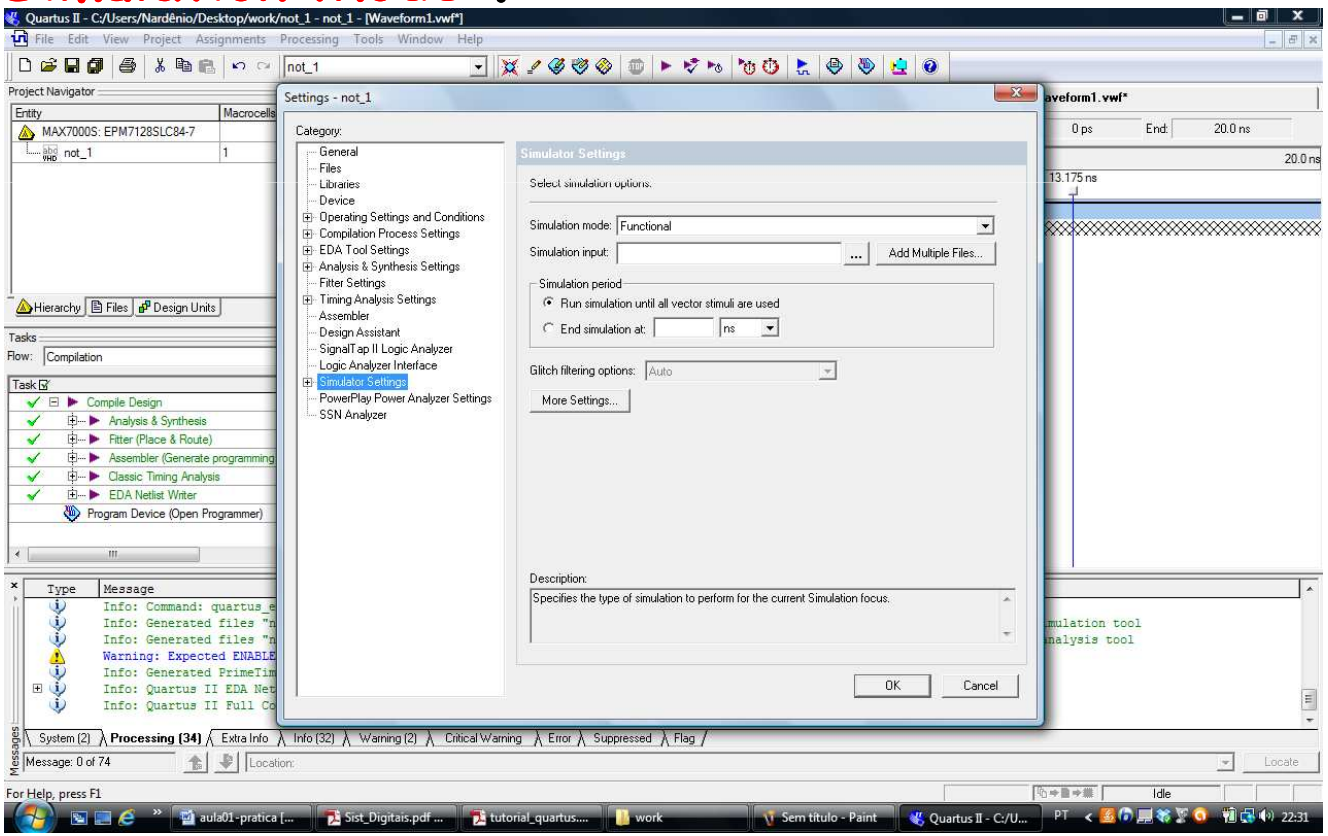
Software Quartus II

19. 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).



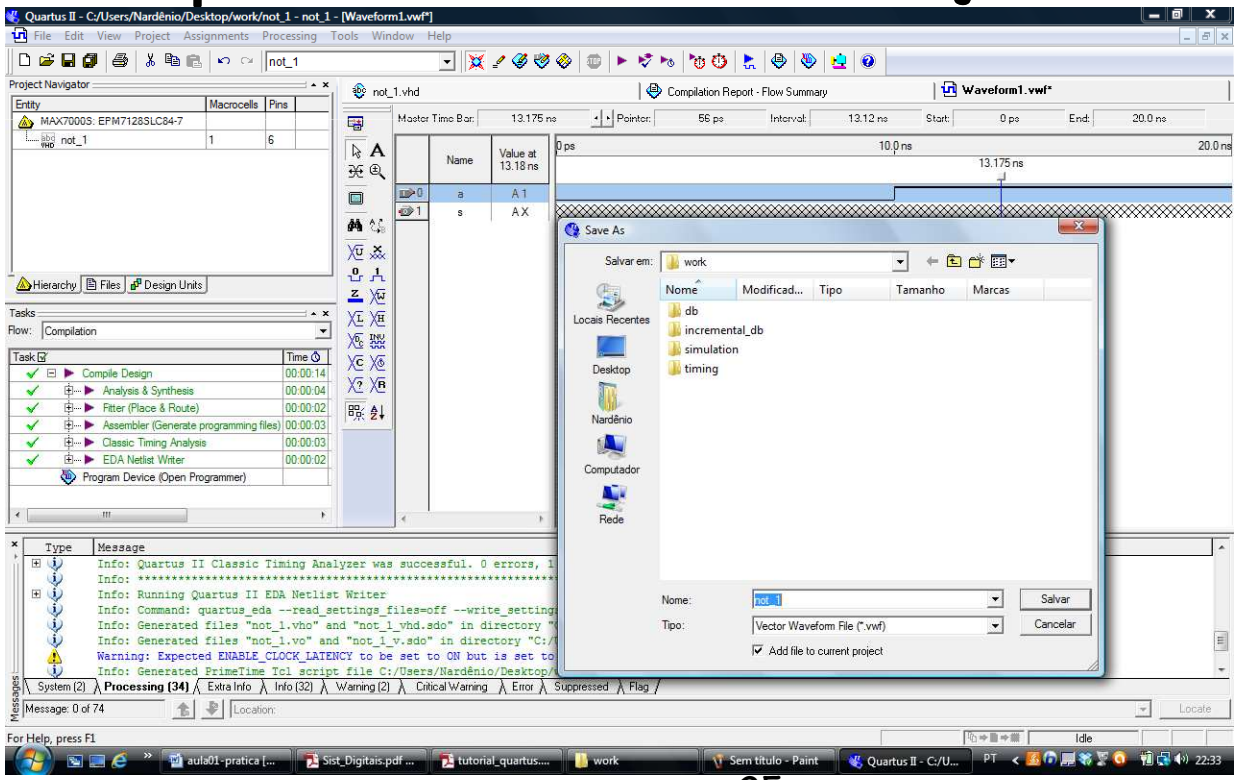
Software Quartus II

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



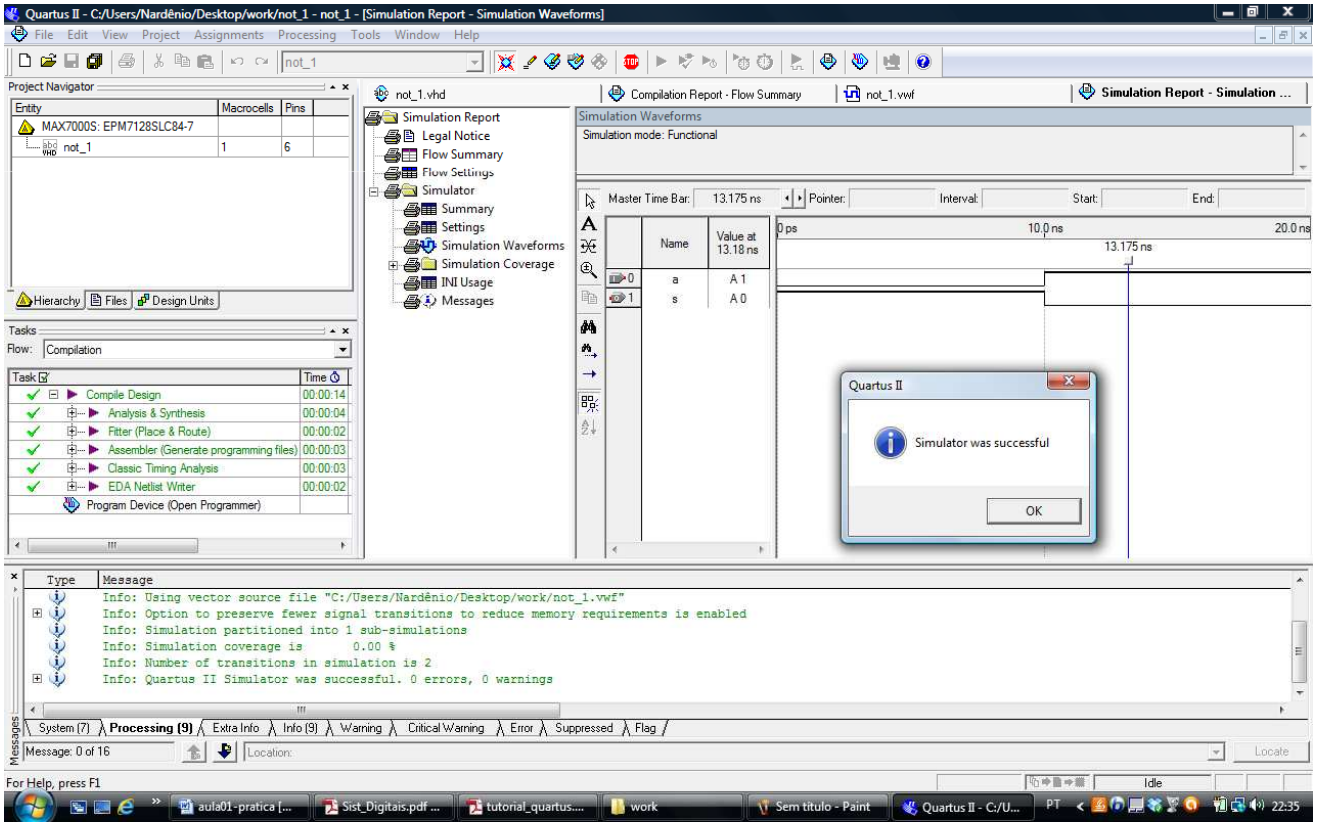
Software Quartus II

21. 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, "mux_cc_case.vwf". Esses passos definem uma simulação funcional.



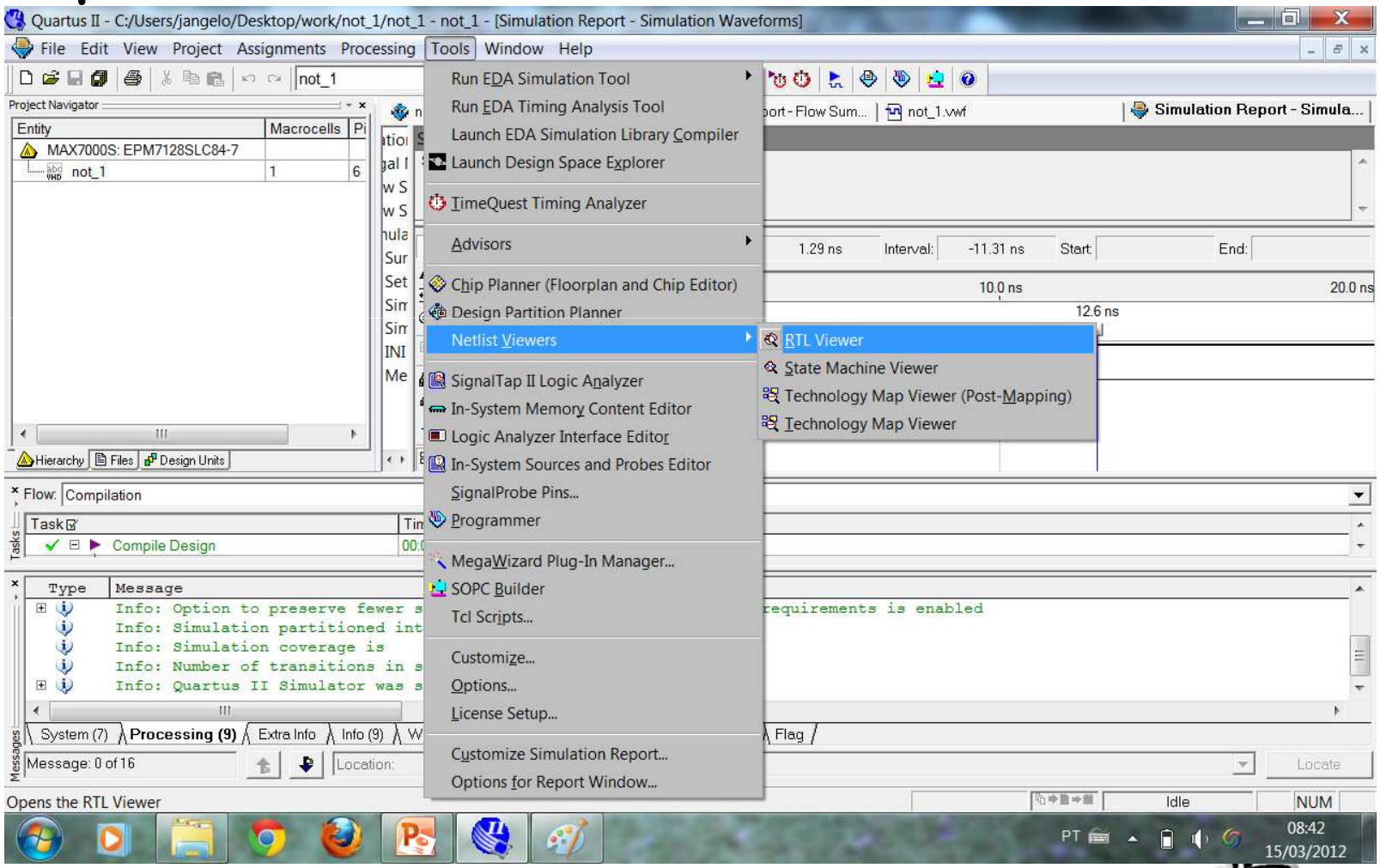
Software Quartus II

22. 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.



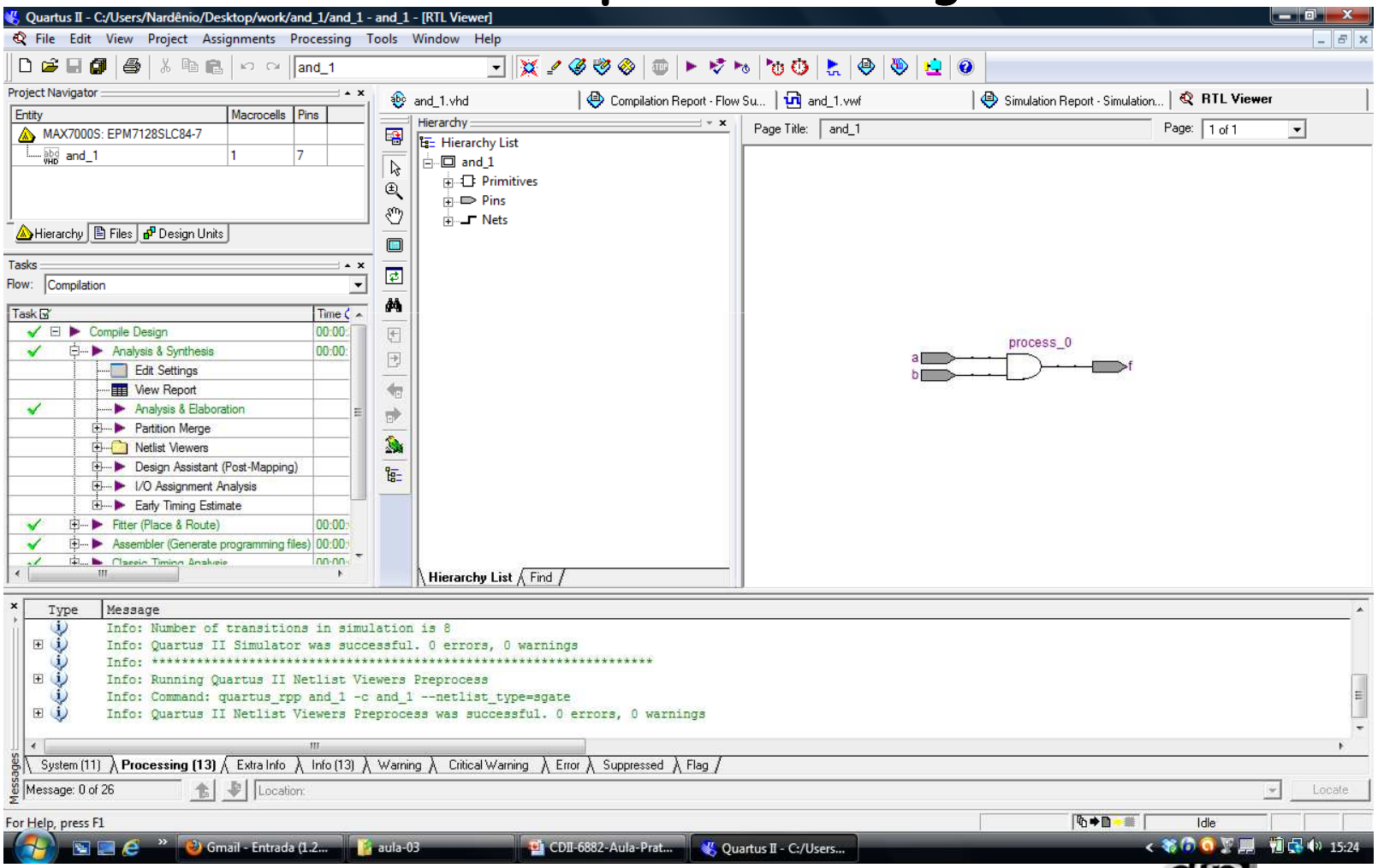
Software Quartus II

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



Software Quartus II

24. O circuito sintetizado a partir do código em VHDL.



Aula de Hoje

Repita os procedimentos para as próximas implementações.

Software Quartus II

Solução 02

```
LIBRARY ieee;          -- Multiplexador 2 x 1 usando comando case when
USE ieee.std_logic_1164.all;
ENTITY mux_cc_case01 IS
    PORT (a, b, s : IN BIT;
                  f : OUT BIT);
END mux_cc_case01;
ARCHITECTURE condicional OF mux_cc_case01 IS
BEGIN
          PROCESS (a, b, s)
          BEGIN
                    CASE s IS
                              WHEN '0' => f <= a;
                              WHEN OTHERS => f <= b;
                    END CASE;
          END PROCESS;
END condicional;
```

Software Quartus II

Solução 03

```
LIBRARY ieee;          -- Multiplexador 2 x 1 usando comando case when
USE ieee.std_logic_1164.all;
ENTITY mux_cc_case02 IS
    PORT (a, b, s : IN STD_LOGIC;
                f : OUT STD_LOGIC);
END mux_cc_case02;
ARCHITECTURE condicional OF mux_cc_case02 IS
BEGIN
          PROCESS (a, b, s)
          BEGIN
                    CASE s IS
                              WHEN '0' => f <= a;
                              WHEN '1' => f <= b;
                              WHEN OTHERS => f <= 'X';
                    END CASE;
          END PROCESS;
END condicional;
```

HDL – Linguagem de Descrição de Hardware

WITH SELECT WHEN

- É um comando concorrente.
- Transfere um valor a um sinal de destino segundo uma relação de opções.
- Todas as condições de seleção devem ser consideradas e elas devem ser mutuamente exclusivas.
- A lista de opções nesta construção não contém uma prioridade.

Sintaxe:

```
WITH expressao_de_escolha SELECT                    -- expressao_de_escolha =
  sinal_destino <= expressao_a WHEN condicao_1,     -- condicao_1
                   expressao_b WHEN condicao_2,     -- condicao_2
                   expressao_c WHEN condicao_3 | condicao_4, -- condicao_3 ou condicao_4
                   expressao_d WHEN condicao_5 TO condicao_9, -- condicao_5 ate condicao_9
                   expressao_e WHEN OTHERS;         -- condicoes restantes
```

- NOTA: O delimitador | equivale a uma operação OU entre as condições de escolha. As palavras reservadas TO e DOWNTO servem para delimitar uma faixa de condições. A palavra reservada OTHERS na última condição serve para agrupar as condições não-relacionadas na lista.

HDL – Linguagem de Descrição de Hardware

Solução 01: "work" > "mux_cc_with"

```
LIBRARY ieee; -- Multiplexador 2 x 1 usando comando with select when
USE ieee.std_logic_1164.all;
ENTITY mux_cc_with IS
    PORT (a, b, s : IN BIT;
                f : OUT BIT);
END mux_cc_with;
ARCHITECTURE condicional OF mux_cc_with IS
BEGIN
          WITH s SELECT
                    f <= a WHEN '0',
                         b WHEN '1';
END condicional;
```

VHDL – Comandos Condicionais

Solução 02: "work" > "mux_cc_with01"

```
LIBRARY ieee; -- Multiplexador 2 x 1 usando comando with select when
USE ieee.std_logic_1164.all;
ENTITY mux_cc_with01 IS
    PORT (a, b, s : IN BIT;
                f : OUT BIT);
END mux_cc_with01;
ARCHITECTURE condicional OF mux_cc_with01 IS
BEGIN
          WITH s SELECT
                    f <= a WHEN '0',
                         b WHEN OTHERS;
END condicional;
```

HDL – Linguagem de Descrição de Hardware

Solução 03: "work" > "mux_cc_with02"

```
LIBRARY ieee; -- Multiplexador 2 x 1 usando comando with select when
USE ieee.std_logic_1164.all;
ENTITY mux_cc_with02 IS
    PORT (a, b, s : IN STD_LOGIC;
                f : OUT STD_LOGIC);
END mux_cc_with02;
ARCHITECTURE condicional OF mux_cc_with02 IS
BEGIN
          WITH s SELECT
                    f <= a WHEN '0',
                         b WHEN '1',
                        'X' WHEN OTHERS;
END condicional;
```

HDL – Linguagem de Descrição de Hardware

WHEN ELSE

- É um comando concorrente
- Restrição de uso dentro de procedimentos, funções e processos
- Transfere o valor de uma expressão para um sinal destino caso uma determinada condição seja satisfeita

Sintaxe

```
sinal_destino <= expressao_1 WHEN condiçao_1 ELSE
                expressao_2 WHEN condiçao_2 ELSE
                expressao_3 WHEN condiçao_3 ELSE
                expressao_4;
```

- Nota: O comando condicional WHEN ELSE é útil para expressar funções lógicas em forma de tabela verdade

HDL – Linguagem de Descrição de Hardware

WHEN ELSE

- Implemente um codificador decimal usando os comandos condicionais WHEN ELSE.
- Considere os nomes das entradas como sendo ch0 até ch9 (para representar chaves de entradas).
- Considere que cada chave de entrada é ativa em nível 0.
- A tabela verdade do codificador decimal é dada a seguir:

HDL – Linguagem de Descrição de Hardware

WHEN ELSE

Tabela Verdade do Codificador Decimal

BCD	saida3	saida2	saida1	saida0
ch0	0	0	0	0
ch1	0	0	0	1
ch2	0	0	1	0
ch3	0	0	1	1
ch4	0	1	0	0
ch5	0	1	0	1
ch6	0	1	1	0
ch7	0	1	1	1
ch8	1	0	0	0
ch9	1	0	0	1

HDL – Linguagem de Descrição de Hardware

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY cod_dec_cc_when IS
            PORT (ch0, ch1, ch2, ch3, ch4, ch5, ch6, ch7, ch8, ch9 : IN BIT;
                        saida  : OUT BIT_VECTOR (3 DOWNTO 0));
END cod_dec_cc_when;
ARCHITECTURE condicional OF cod_dec_cc_when IS
BEGIN
     saida <= "0000" WHEN ch0 = '0' ELSE
               "0001" WHEN ch1 = '0' ELSE
               "0010" WHEN ch2 = '0' ELSE
               "0011" WHEN ch3 = '0' ELSE
               "0100" WHEN ch4 = '0' ELSE
               "0101" WHEN ch5 = '0' ELSE
               "0110" WHEN ch6 = '0' ELSE
               "0111" WHEN ch7 = '0' ELSE
               "1000" WHEN ch8 = '0' ELSE
               "1001";
END condicional;
```