# 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

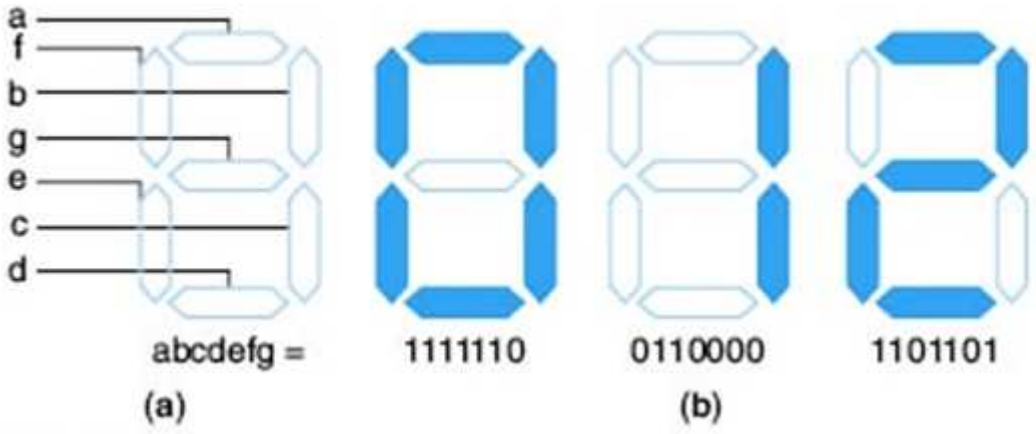
**Projeto e Simulação dos seguintes circuitos:**

- **Decodificador BCD para display de 7 segmentos**
  - ✓ Comando WITH SELECT WHEN
- **Subtrator de 2 bits**
  - ✓ Comando FOR LOOP
  - ✓ Comando WHILE LOOP

# HDL – Linguagem de Descrição de Hardware

## Decodificador BCD para display de 7 segmentos

Considere o display de 7 segmentos mostrado na figura e a tabela verdade a seguir. Projete o decodificador do código BCD para o display de 7 segmentos.

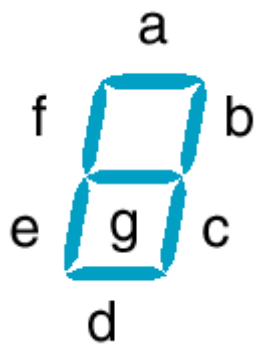




# HDL – Linguagem de Descrição de Hardware

## Decodificador BCD para display de 7 segmentos

Tabela verdade para o decodificador do código BCD para o display de 7 segmentos.

| BCD | Display | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|---|
| 0000 | | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0001 | | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0010 | | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0011 | | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0100 | | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0101 | | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0110 | | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0111 | | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1000 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1001 | | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| ... | | | | | | | | |
| 1111 | | X | X | X | X | X | X | X |



# HDL – Linguagem de Descrição de Hardware

## Decodificador BCD para display de 7 segmentos

Obtenção das expressões booleanas do decodificador BCD para display de 7 segmentos

$a = A + C + BD + \overline{B}\,\overline{D}$

$b = \overline{B} + \overline{C}\,\overline{D} + CD$

$c = B + \overline{C} + D$

$d = A + \overline{B}\,\overline{D} + \overline{B}C + C\overline{D} + \overline{B}CD$

$e = \overline{B}\,\overline{D} + C\overline{D}$

$f = A + \overline{C}\,\overline{D} + B\overline{C} + B\overline{D}$

$g = A + B\overline{C} + \overline{B}C + C\overline{D}$

# HDL – Linguagem de Descrição de Hardware

## Estrutura Básica de um Código em VHDL

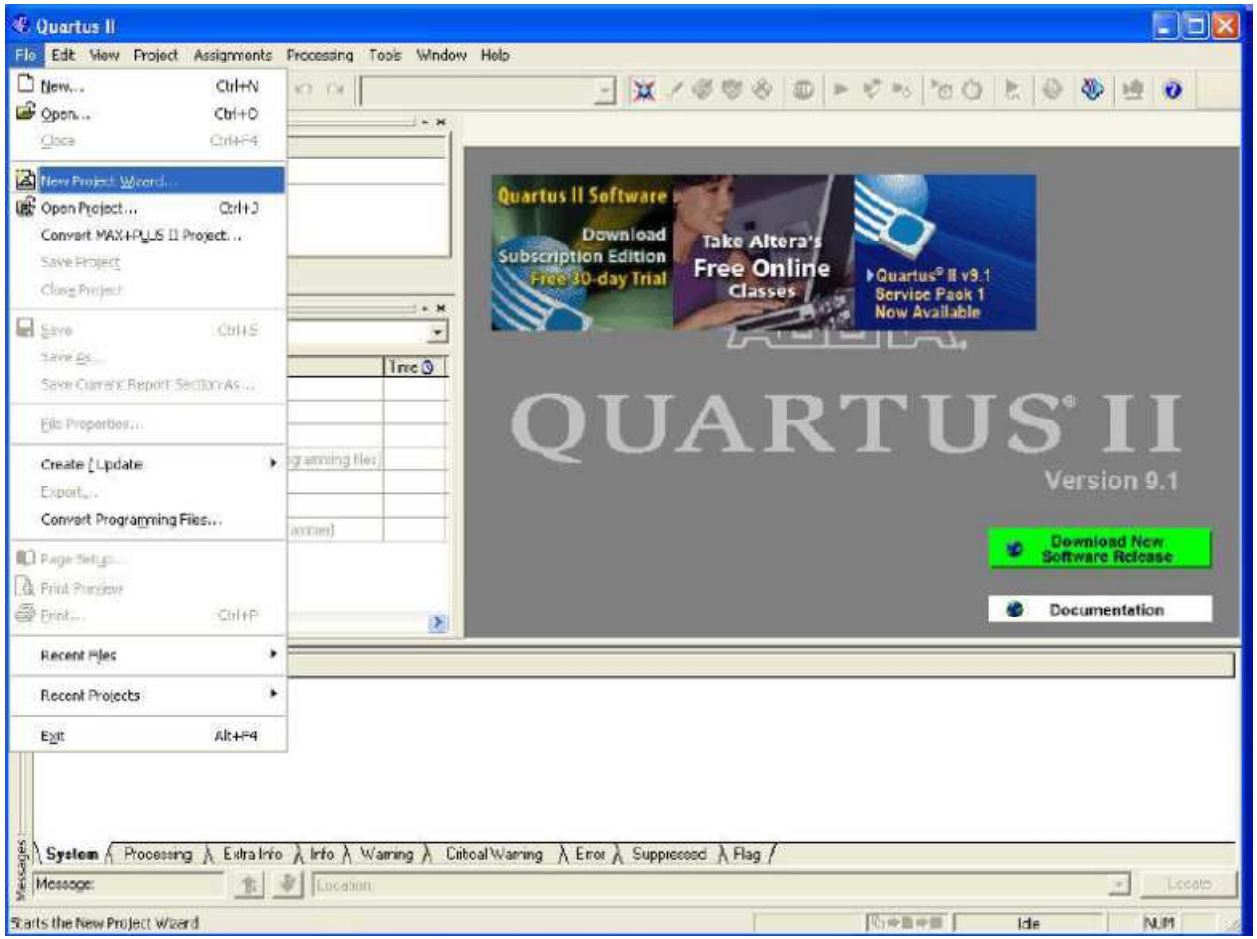
| | |
|---|---|
| **LIBRARY IEEE;**<br>**USE IEEE.STD_LOGIC_1164.all;**<br>**USE IEEE.STD_LOGIC_UNSIGNED.all;** | LIBRARY (PACOTES) |
| **ENTITY** exemplo **IS**<br>**PORT** (<br><descrição dos pinos de I/O><br>);<br>**END** exemplo; | ENTITY (PINOS DE I/O) |
| **ARCHITECTURE** teste **OF** exemplo **IS**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

# Software Quartus II

1. Crie diretório ou pasta "work" na área de trabalho.
2. Crie os seguintes subdiretórios dentro do diretório "work":
   a) "deco_bcd_7seg"
   b) "sub_2bits_for"
   c) "sub_2bits_while"
3. Inicialize o Software Quartus II

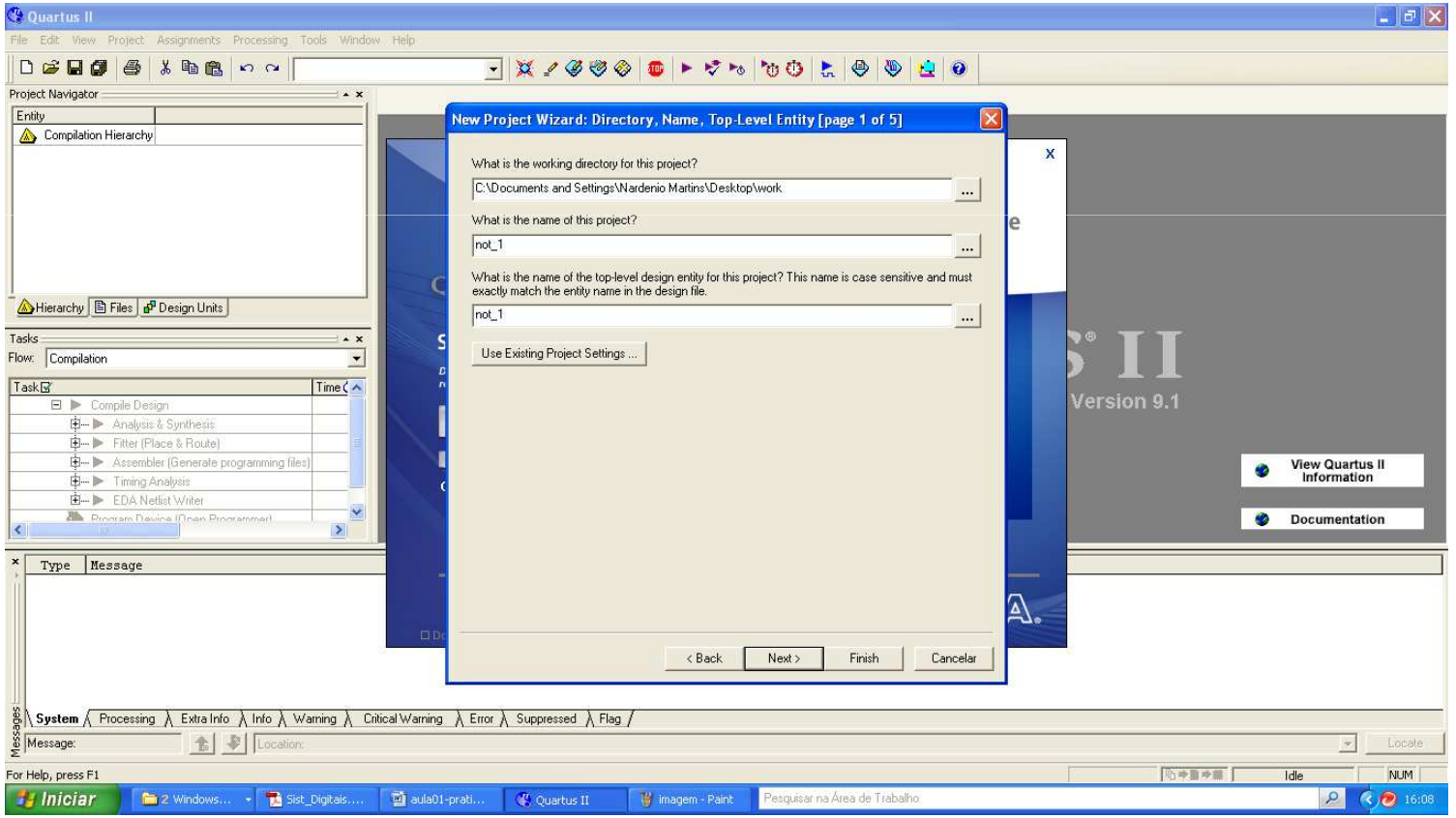
# Software Quartus II

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



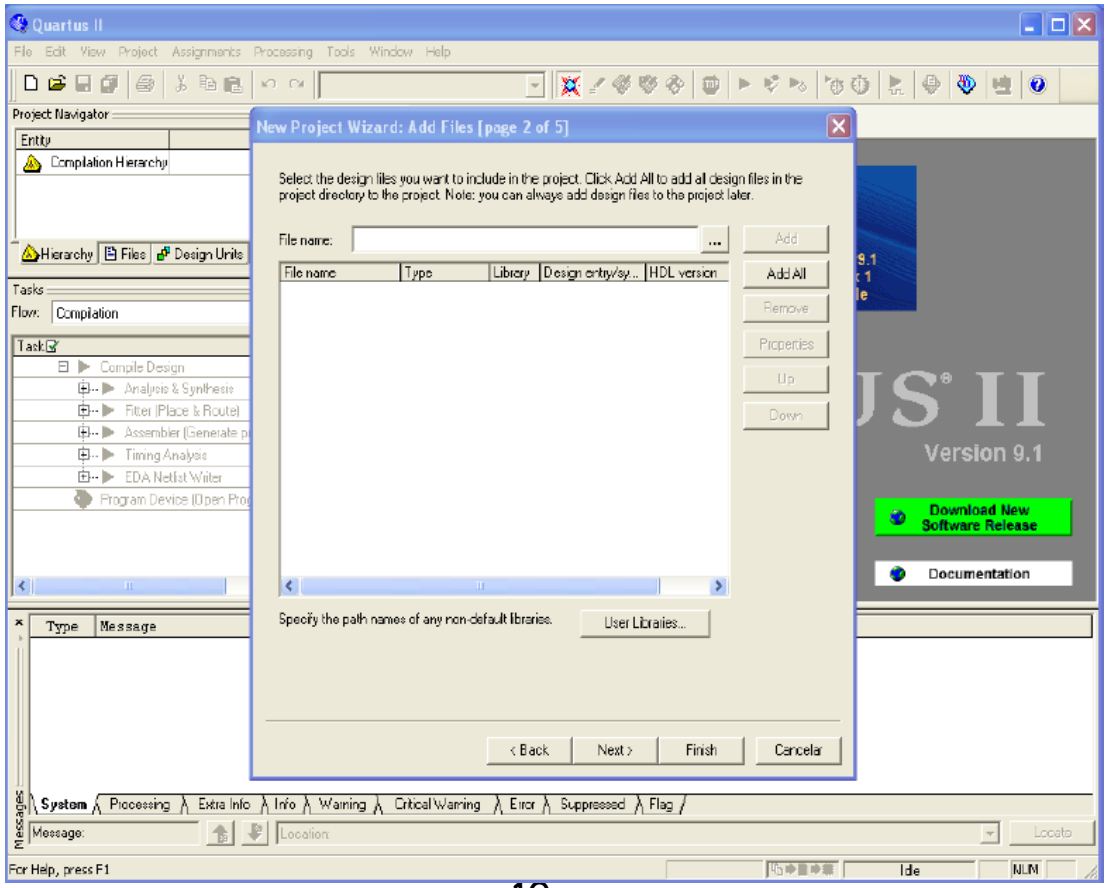
# Software Quartus II

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 → "deco_bcd_7seg".



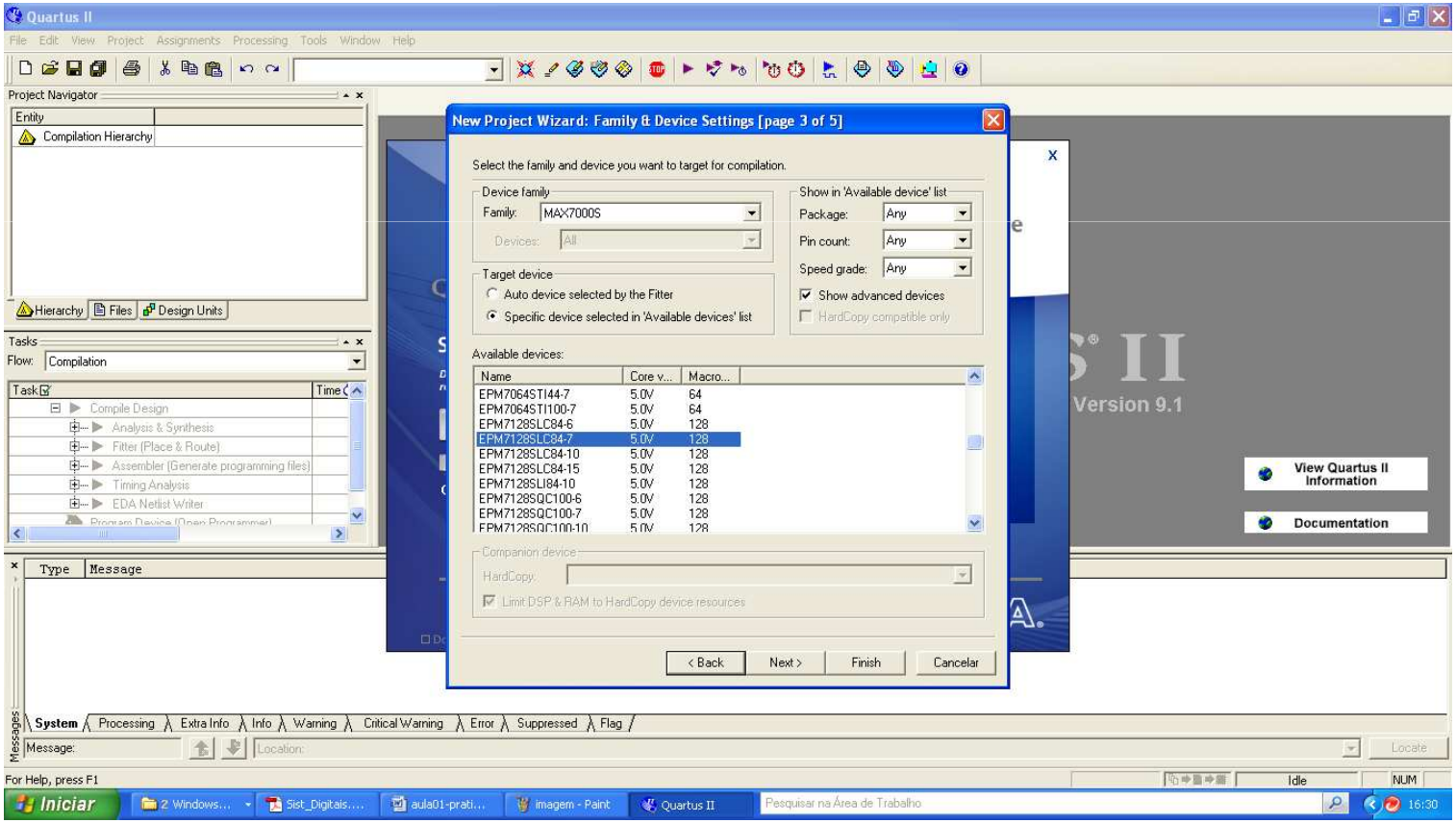
# Software Quartus II

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



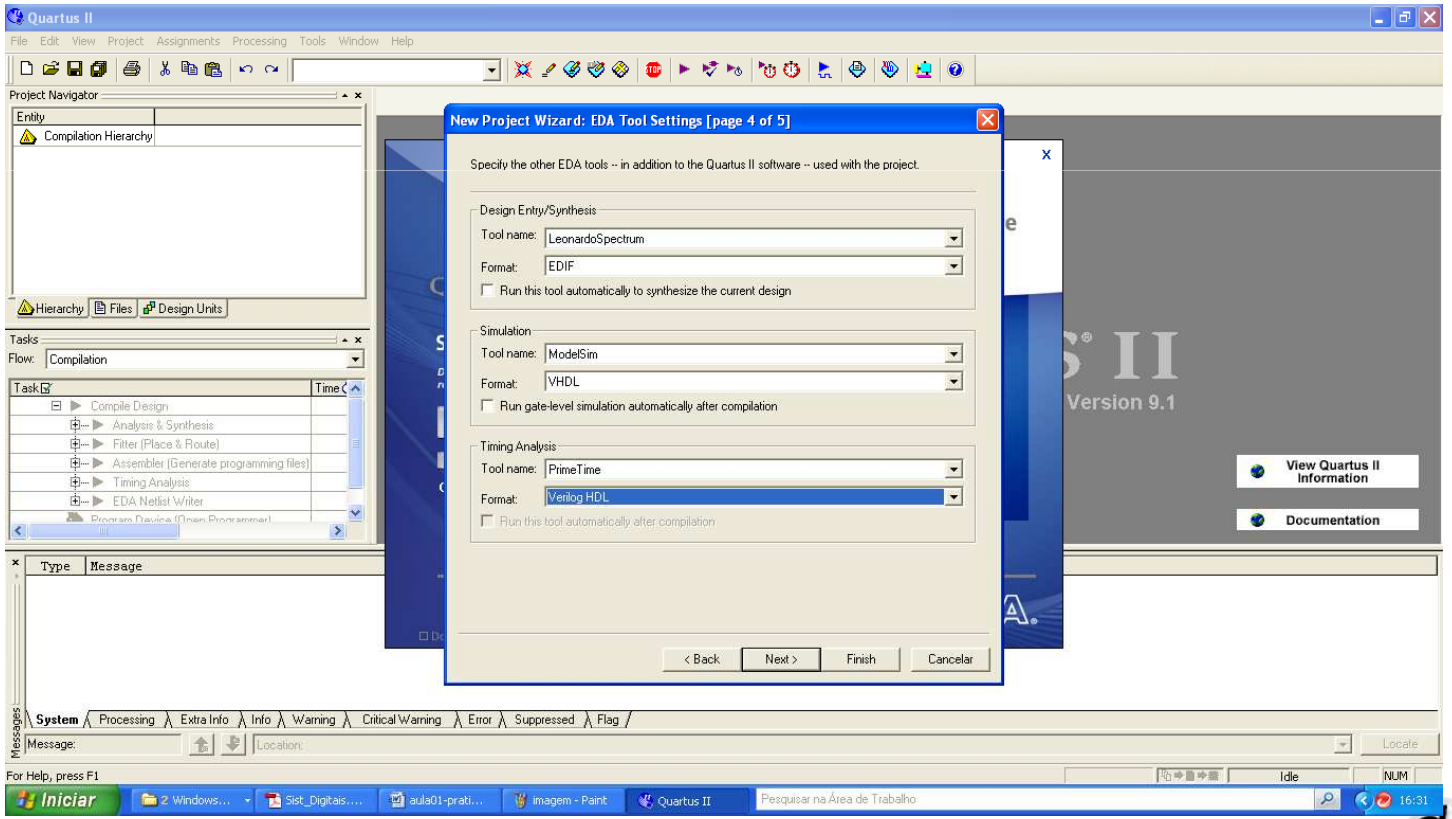
# Software Quartus II

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



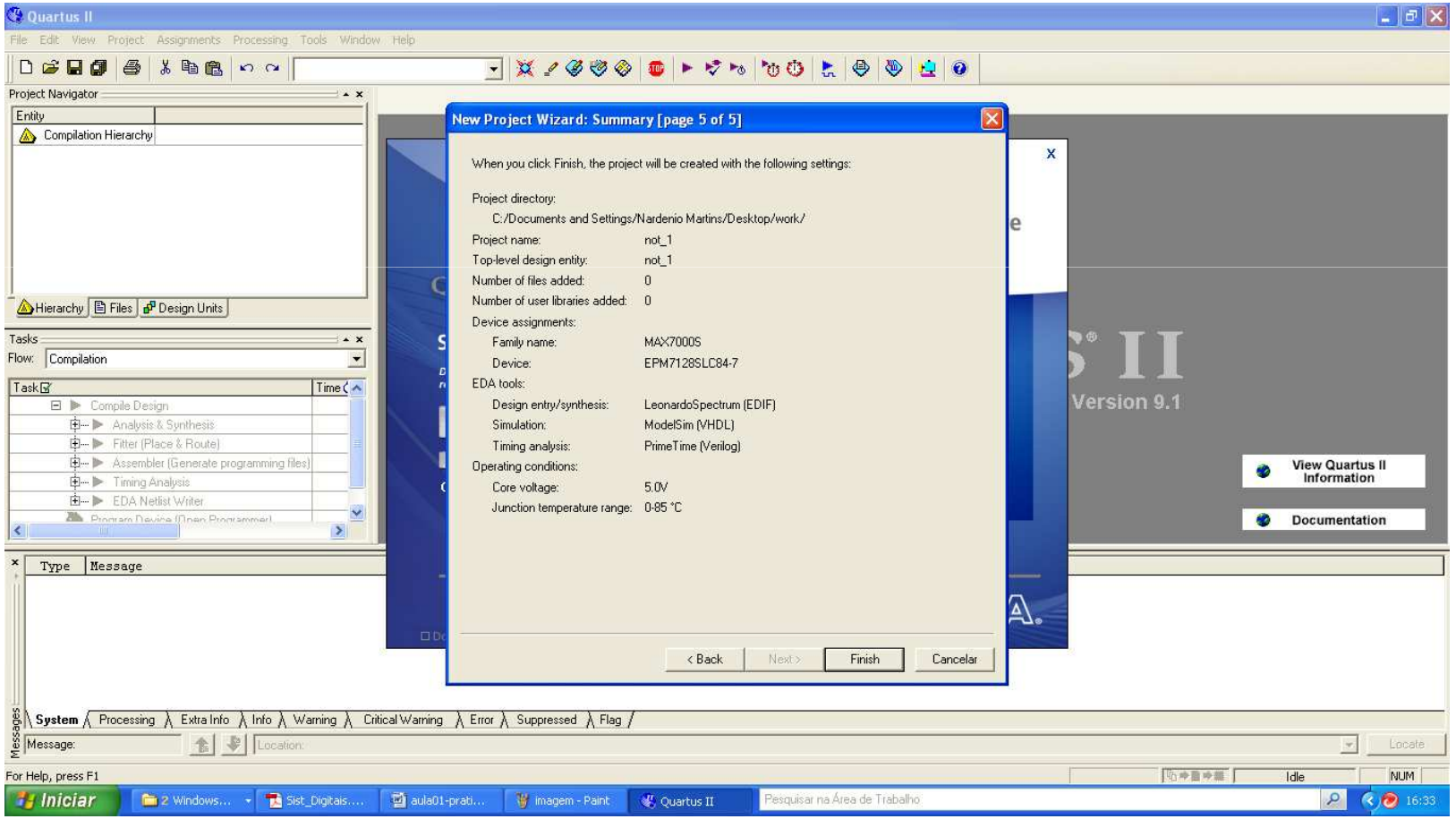
# Software Quartus II

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.



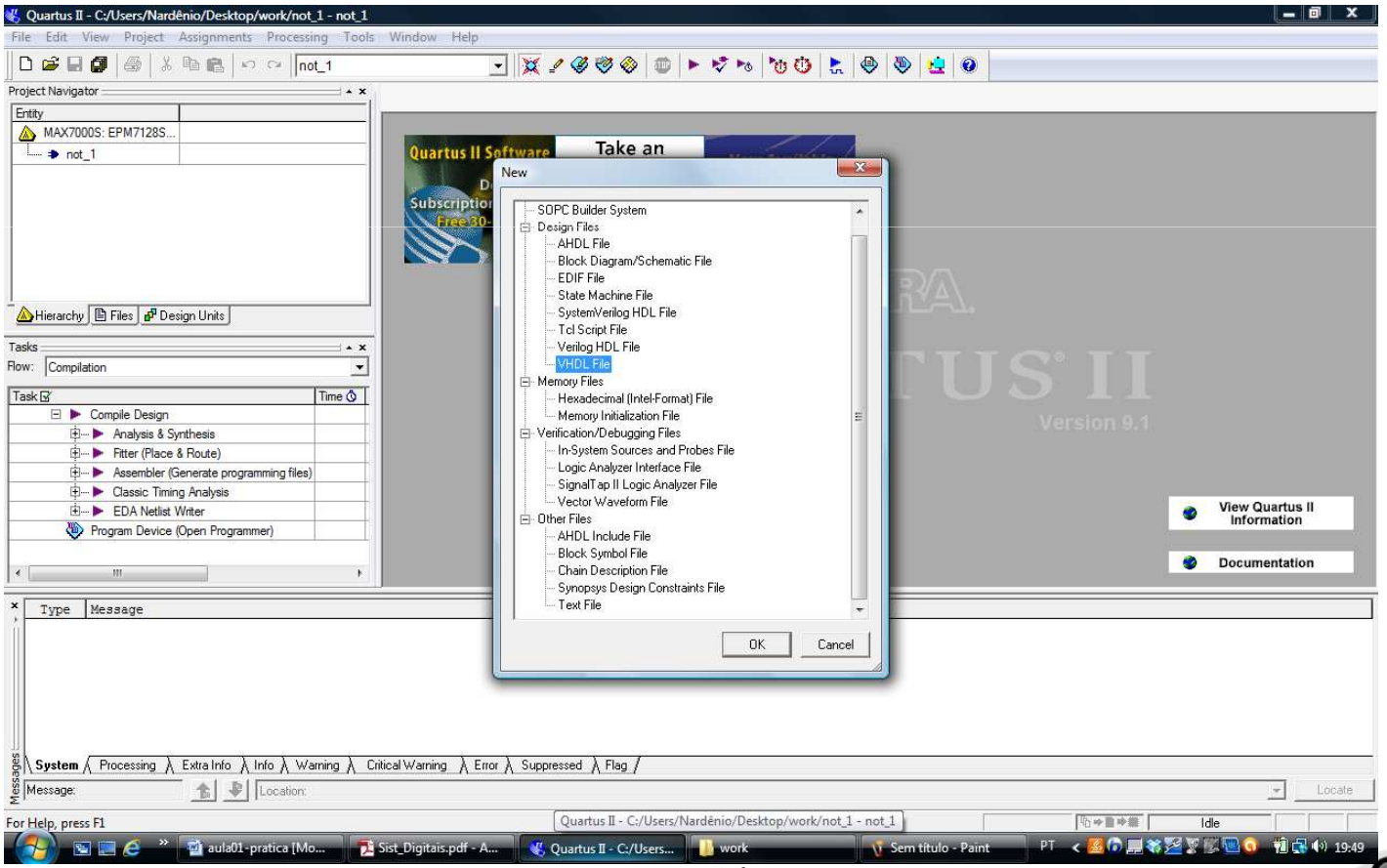
# Software Quartus II

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

## 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.

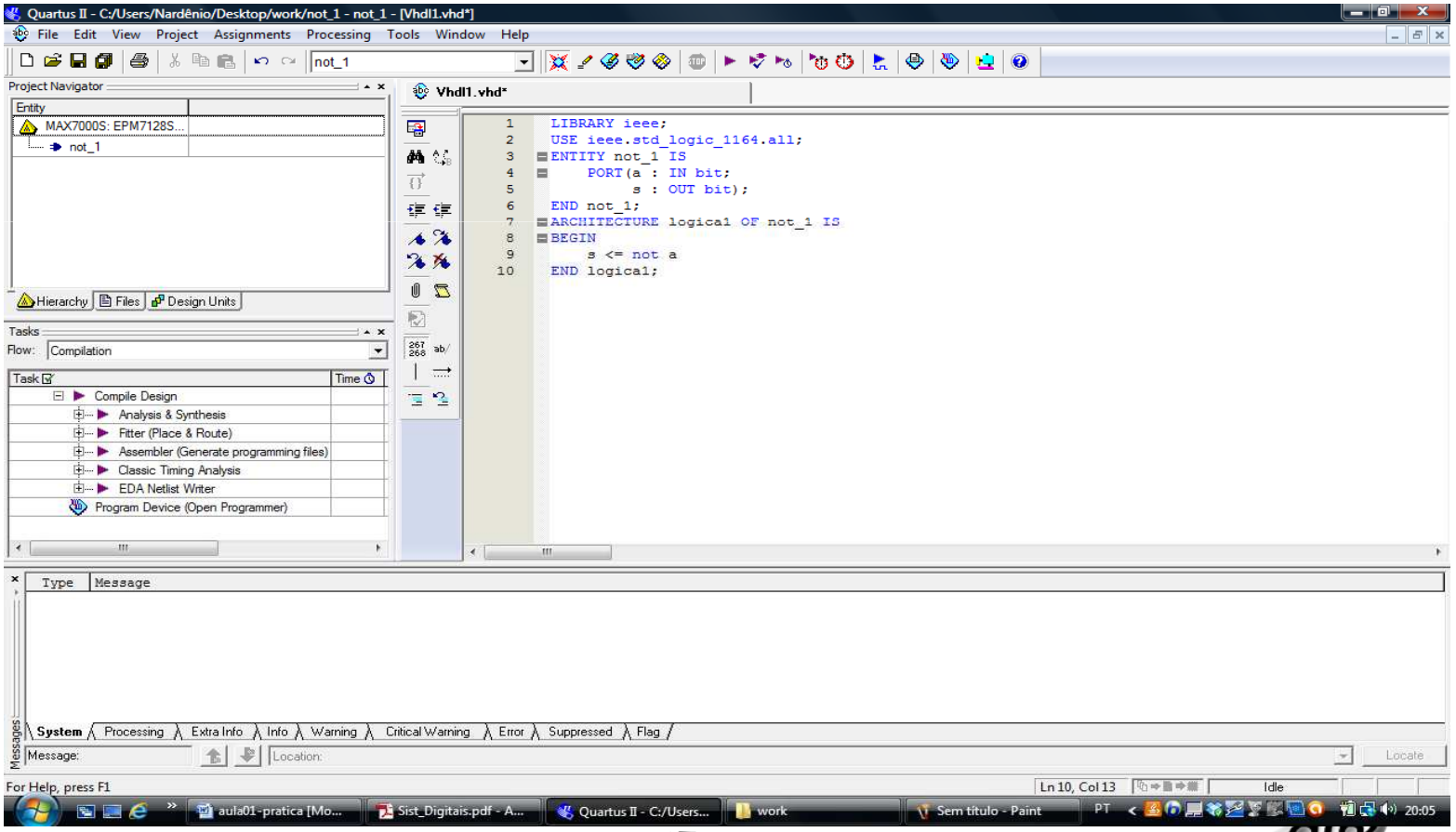
# Software Quartus II

## 11. Escreva o código em VHDL.

```
LIBRARY ieee;          -- Decodificador BCD para display de 7 Segmentos
USE ieee.std_logic_1164.all;
ENTITY deco_bcd_7seg IS
          PORT( bcd: IN BIT_VECTOR(3 DOWNTO 0);
                segmentos: OUT BIT_VECTOR(6 DOWNTO 0));
END deco_bcd_7seg;
ARCHITECTURE teste OF deco_bcd_7seg IS
BEGIN
WITH bcd SELECT
          segmentos <= "1111110" WHEN "0000",
                       "0110000" WHEN "0001",
                       "1101101" WHEN "0010",
                       "1111001" WHEN "0011",
                       "0110011" WHEN "0100",
                       "1011011" WHEN "0101",
                       "1011111" WHEN "0110",
                       "1110000" WHEN "0111",
                       "1111111" WHEN "1000",
                       "1111011" WHEN "1001",
                       "1111110" WHEN OTHERS;
END teste;
```

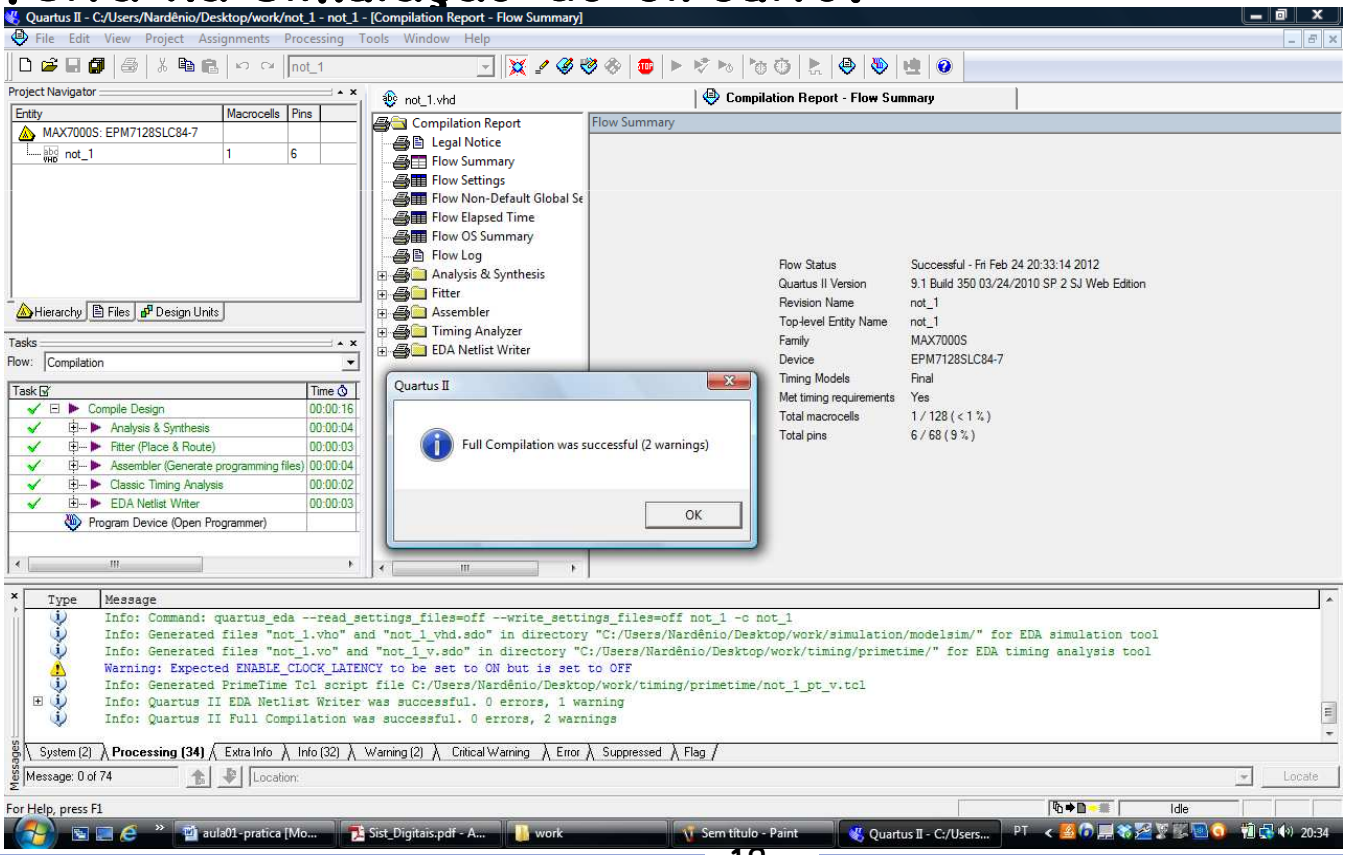
# Software Quartus II

12. Salve o código em VHDL com extensão "deco_bcd_7seg.vhd" na pasta ou subdiretório "deco_bcd_7seg".



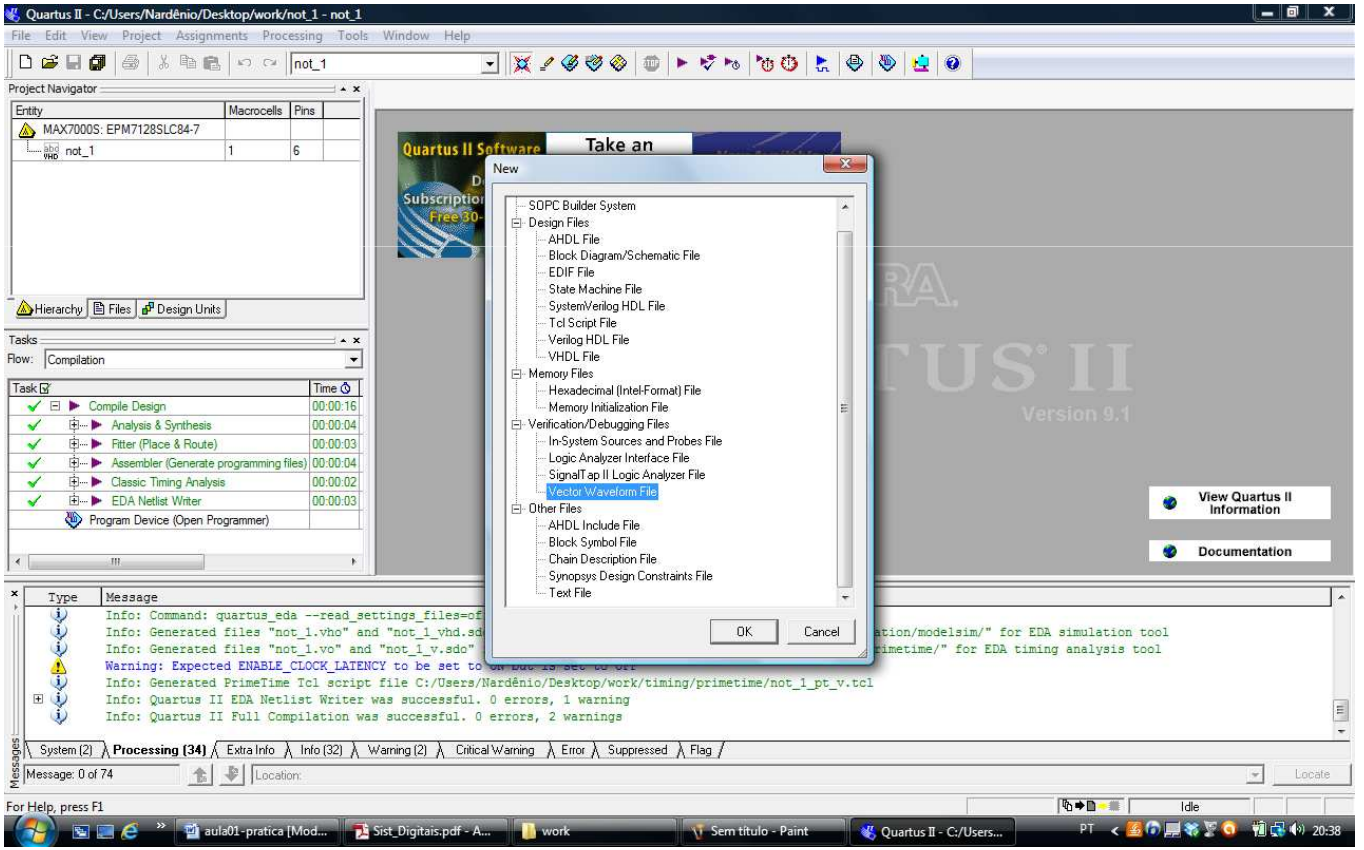
# Software Quartus II

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.



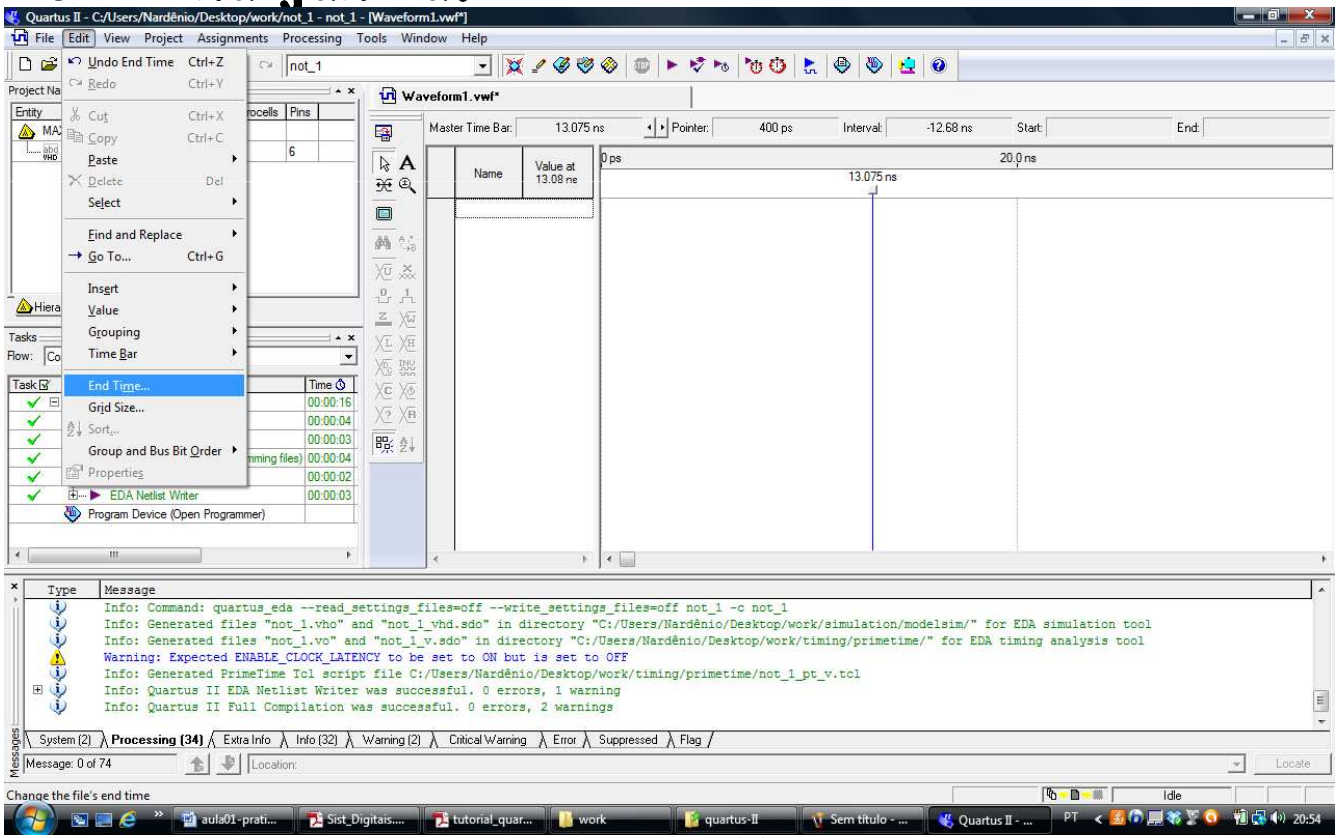
# Software Quartus II

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



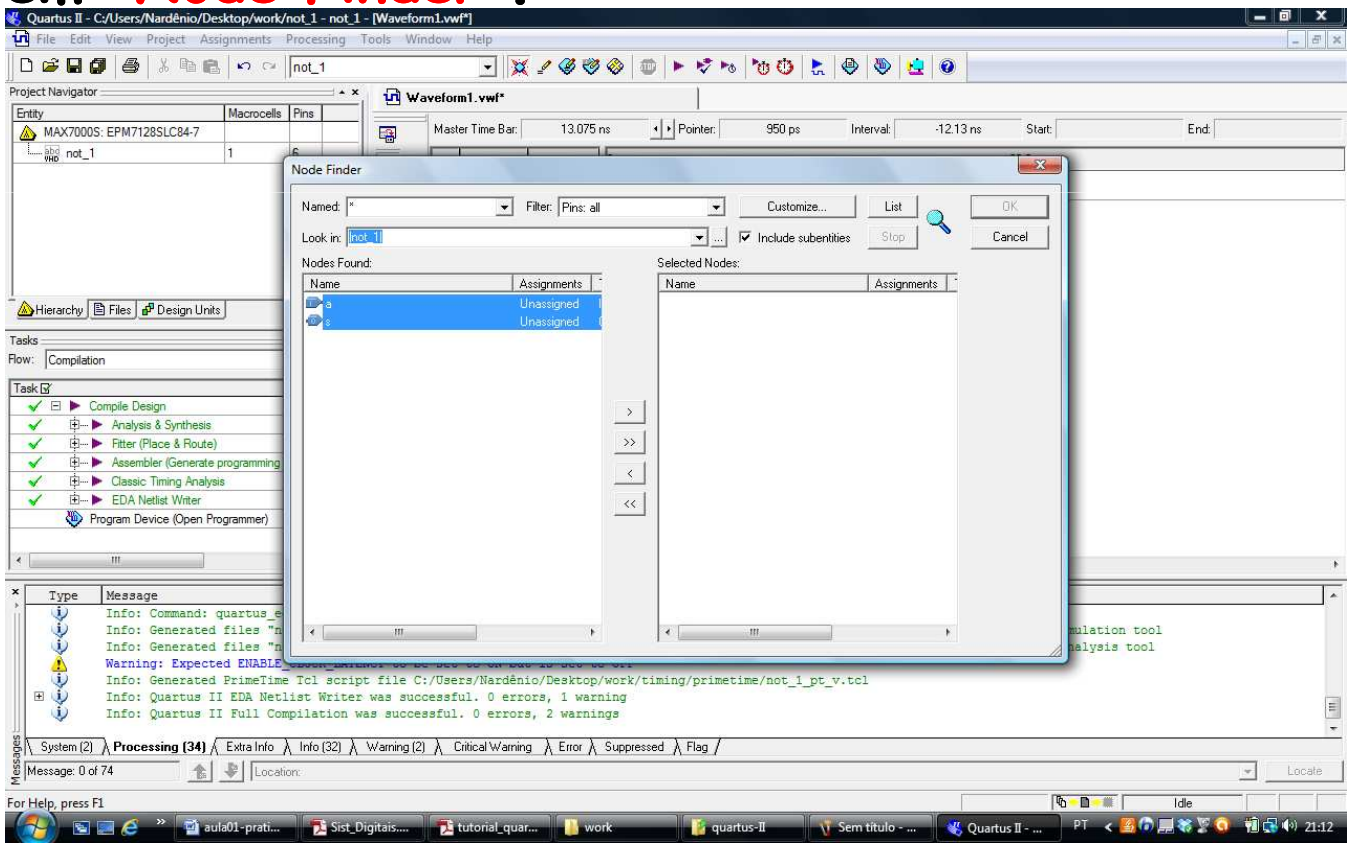
# Software Quartus II

15. Ajuste o tempo de simulação: "Edit > End Time" e coloque 100 ns para simulação. Clique "View > Fit in Window" para que todo o tempo de simulação fique visível na janela.



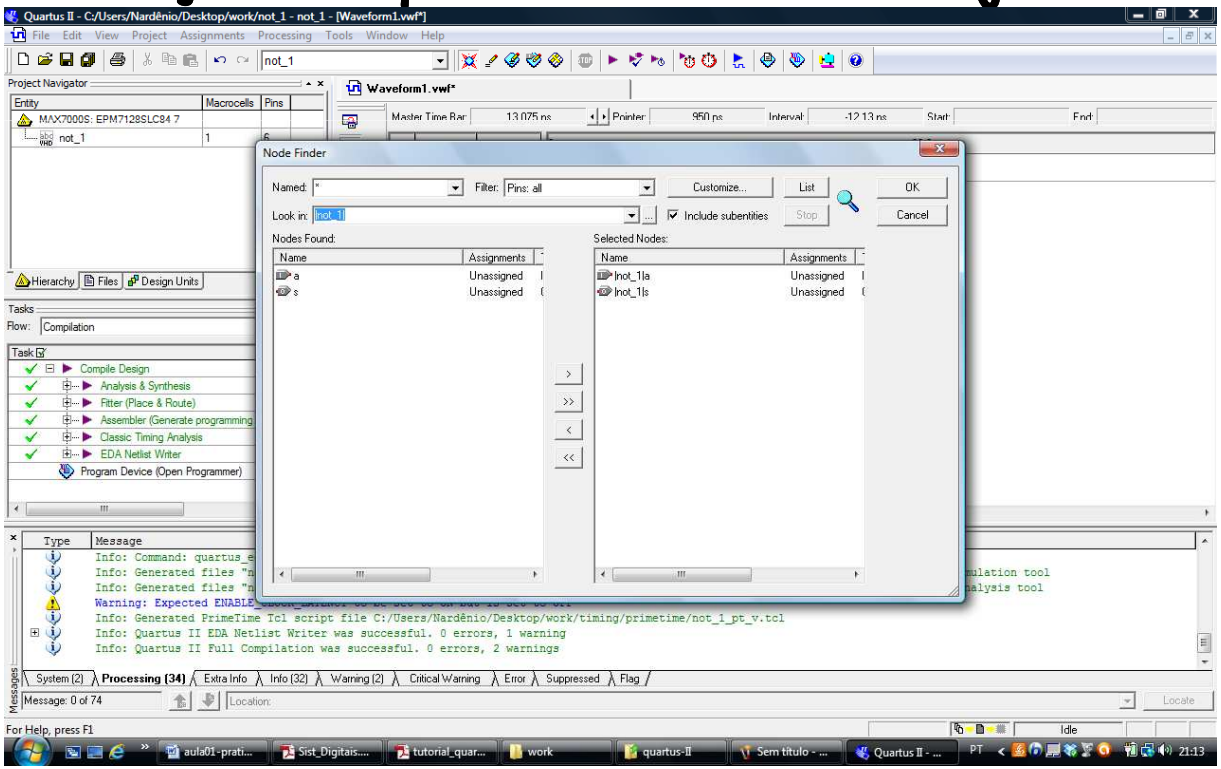
# Software Quartus II

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".



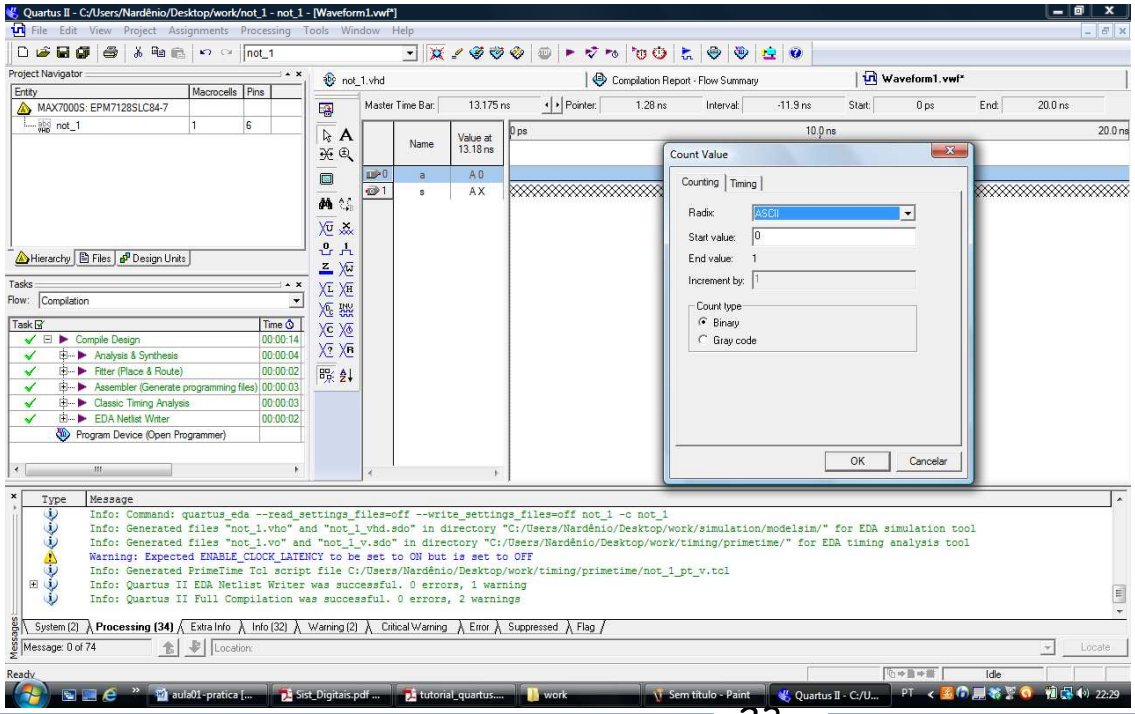
# Software Quartus II

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.



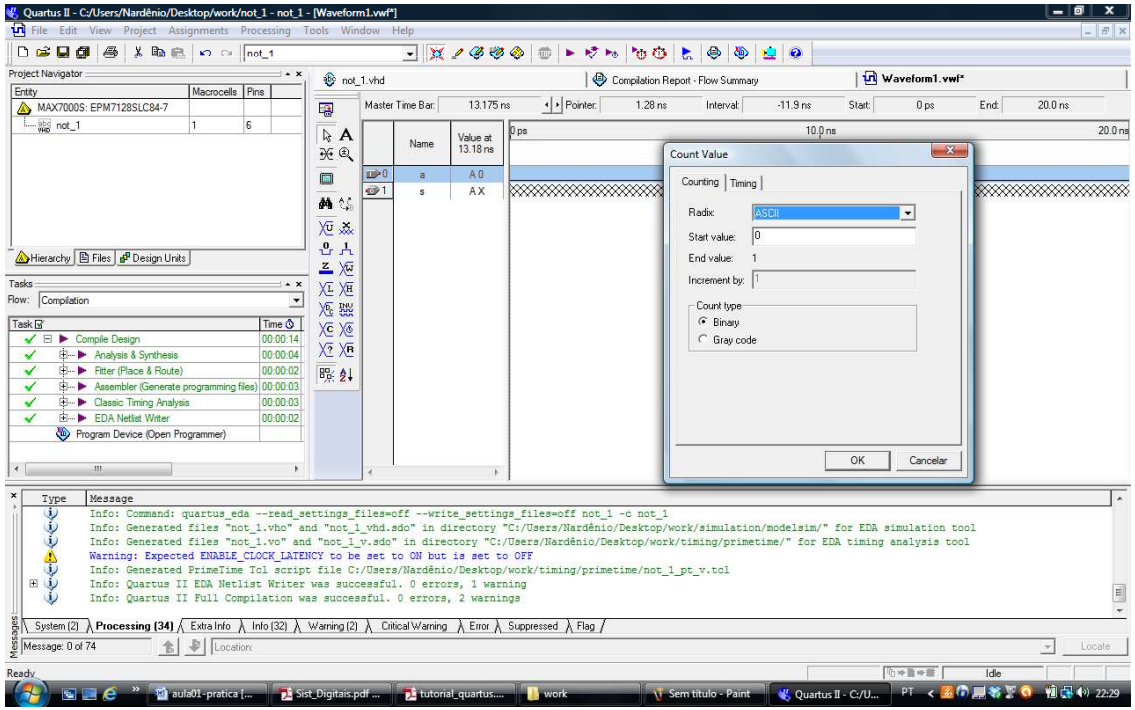
# Software Quartus II

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").



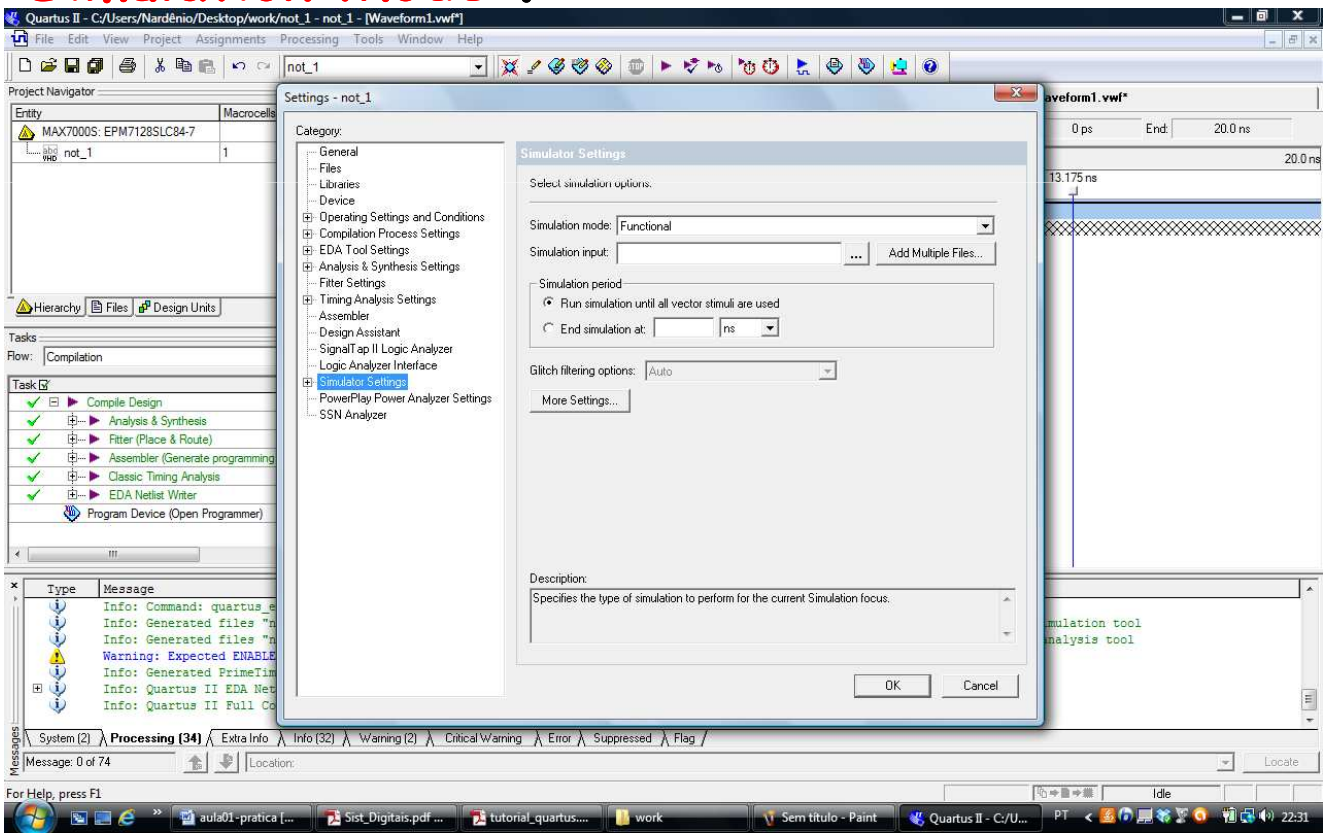
# Software Quartus II

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, [9] (na realidade, pode-se ver que os bits de entrada como quatro entradas de 1 *bit*, que pode assumir, portanto, valores de 0000 a 1001).



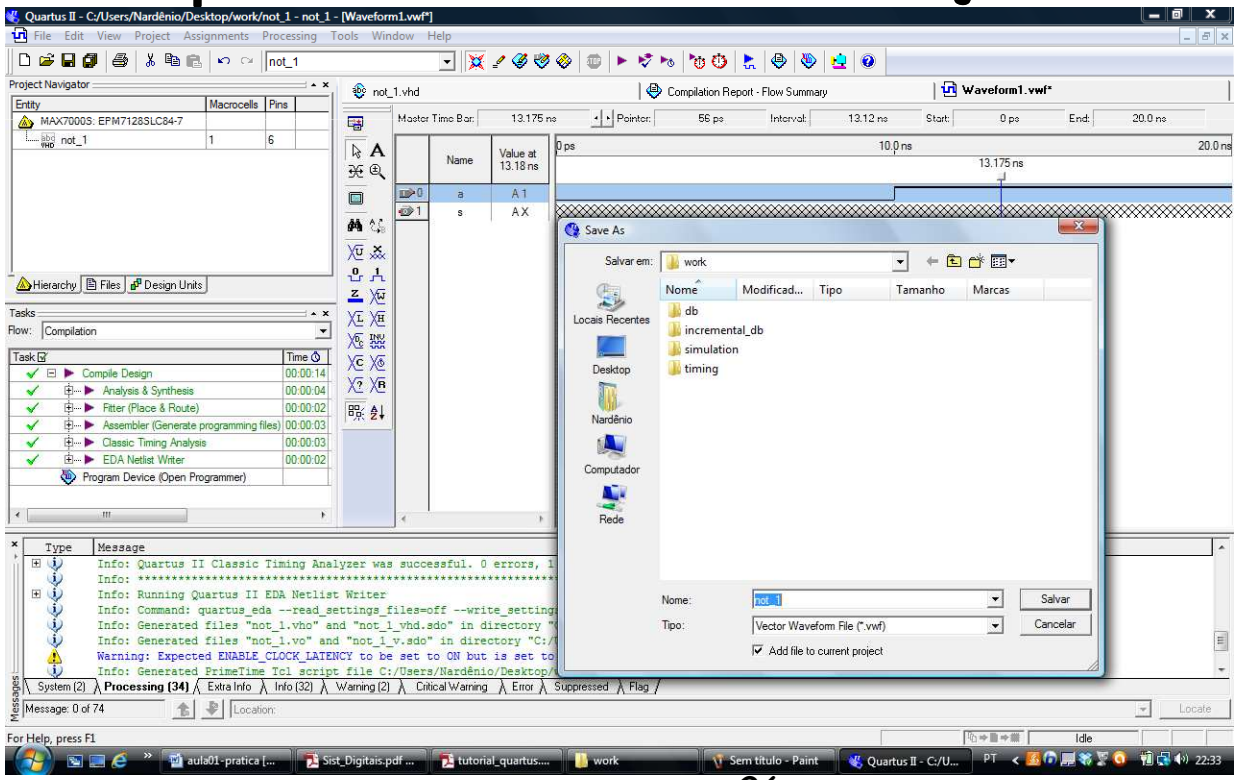
# Software Quartus II

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



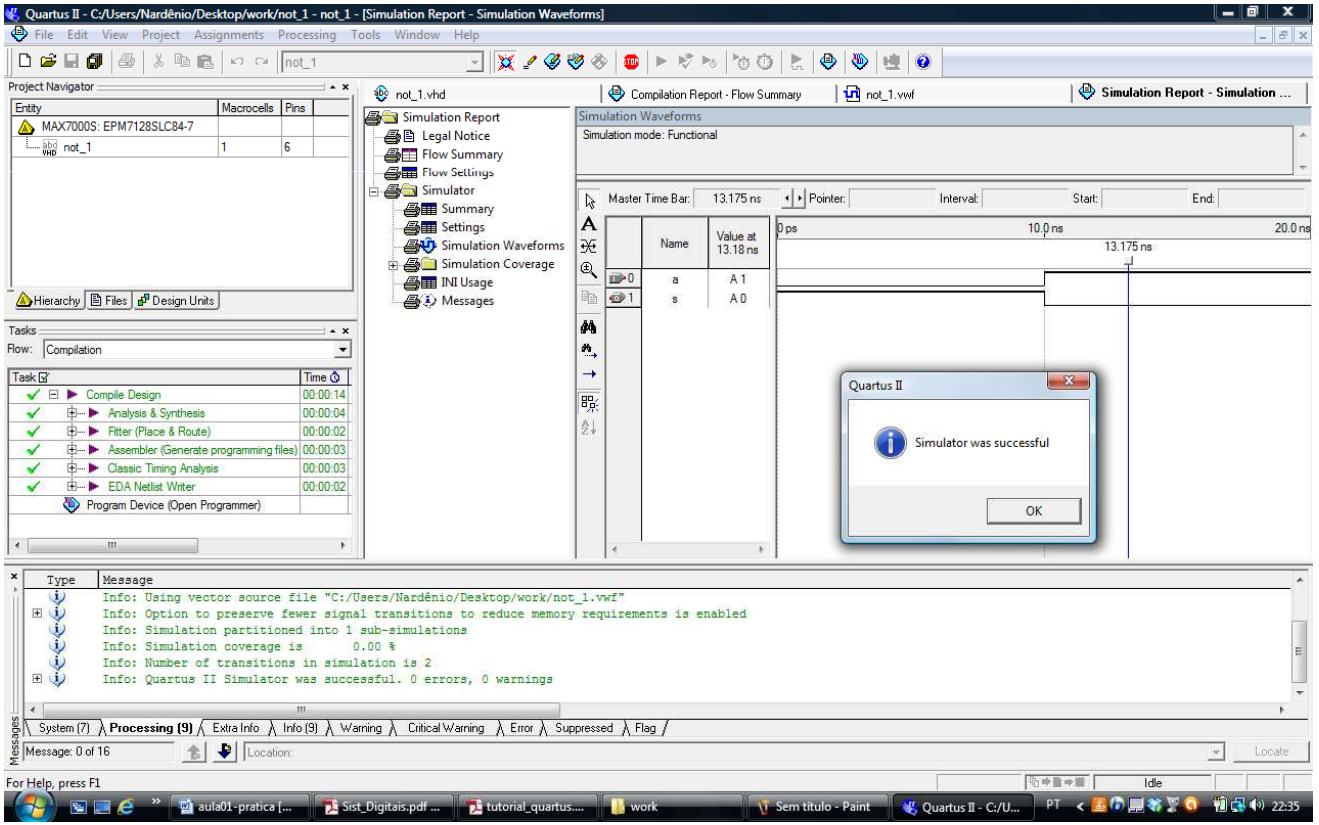
# Software Quartus II

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, "deco_bcd_7seg.vwf". Esses passos definem uma simulação funcional.



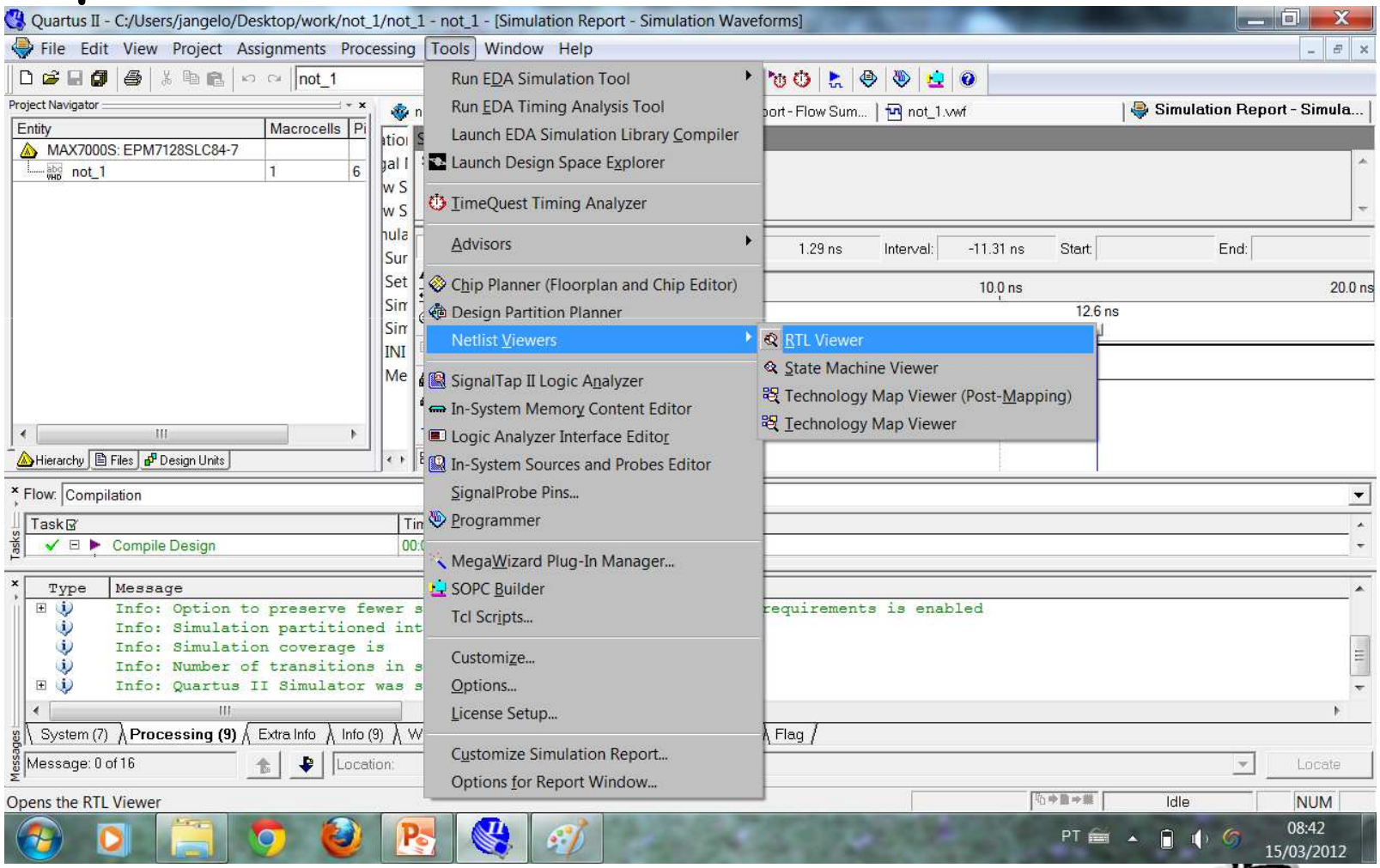
# Software Quartus II

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.



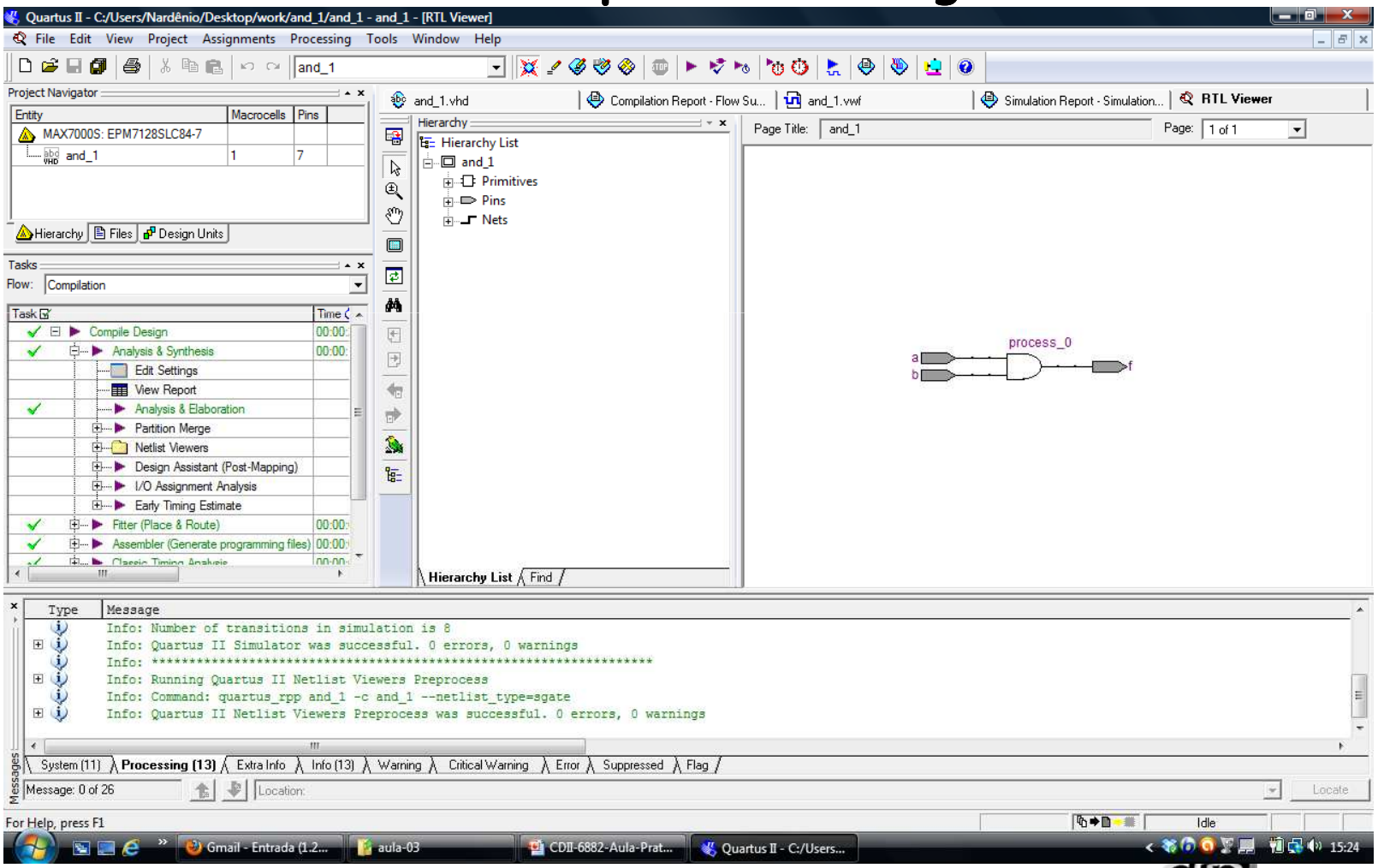
# Software Quartus II

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



# Software Quartus II

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



# Aula de Hoje

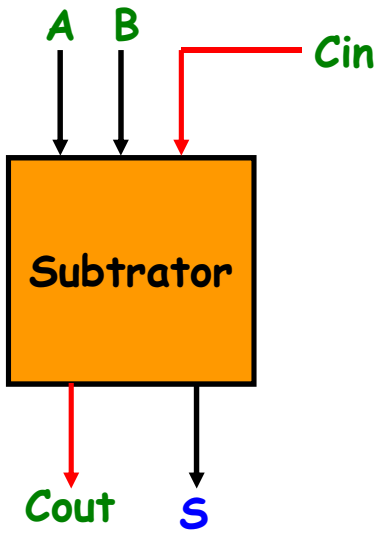
Repita os procedimentos para as próximas implementações.

# HDL – Linguagem de Descrição de Hardware

## Subtrator Completo

Entradas | Saídas

| A | B | $C_{in}$ | S | $C_{out}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |




$$S = A \oplus B \oplus C_{in}$$

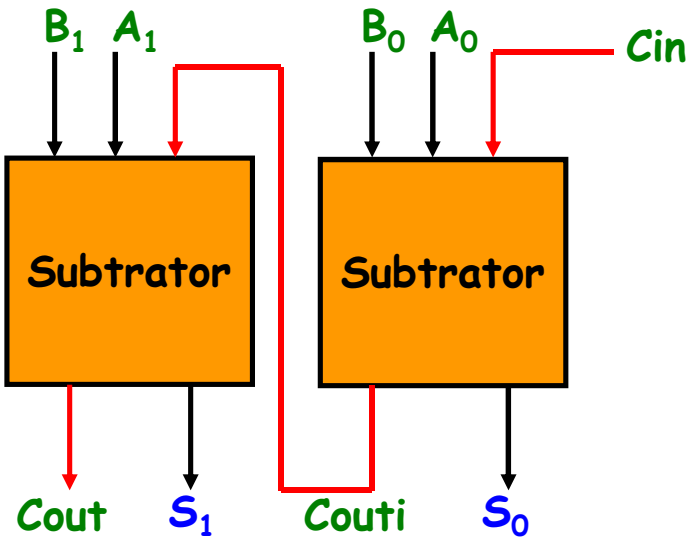
$$C_{out} = \overline{A}B + BC_{in} + \overline{A}C_{in}$$

ou

$$C_{out} = = \overline{(A \oplus B)}\, Cin + \overline{A}B$$

# HDL – Linguagem de Descrição de Hardware

## Subtrator Completo



# HDL – Linguagem de Descrição de Hardware

## FOR LOOP

- Permite a repetição de instruções uma quantidade de vezes preestabelecida
- Restrição de uso dentro de procedimentos, funções e processos
- Um contador vai sendo incrementado ou decrementado a cada iteração até atingir um valor limite.
- Contador não pode ser alterado com operações de atribuição

### Sintaxe

```
FOR contador IN valor_inicial TO|DOWNTO valor_final LOOP
                    comandos
                    ...
END LOOP;
```

# HDL – Linguagem de Descrição de Hardware

**Solução**

- **Exercício 01:**

```
ENTITY sub_2bits IS
  GENERIC (n          : INTEGER  := 2);                    -- numero de bits
  PORT     (A, B      : IN  BIT_VECTOR (n-1 DOWNTO 0);     -- entradas do subtrator
            Cin       : IN  BIT;                           -- empresta-1
            S         : OUT BIT_VECTOR (n-1 DOWNTO 0);     -- saida
            Cout      : OUT BIT);                          -- vai-1
END sub_2bits ;

ARCHITECTURE logica OF sub_2bits IS
BEGIN
            PROCESS (A, B, Cin)
            VARIABLE Couti : BIT_VECTOR (n DOWNTO 0);      -- vai-1 interno
            BEGIN
                        Couti(0) := Cin;
                        FOR i IN 0 TO n-1 LOOP
                        S(i)   <=  A(i) XOR B(i)  XOR Couti(i);
                        Couti(i+1) := ((not A(i)) AND B(i)) OR ((not A(i)) AND Couti(i)) OR (B(i) AND Couti(i));
                        END LOOP;
                        Cout   <= Couti(n);
            END PROCESS;
END logica;
```

# HDL – Linguagem de Descrição de Hardware

## WHILE LOOP

- Permite a repetição de instruções se uma condição for verdadeira
- A iteração termina se a condição for falsa
- Restrição de uso dentro de procedimentos, funções e processos

## Sintaxe

```
WHILE condição LOOP
                    comandos
                    ...
END LOOP;
```

# HDL – Linguagem de Descrição de Hardware

- **Exercício 02:** **Solução**

```
ENTITY sub_2bits_while IS
  GENERIC (n          : INTEGER  := 2);                    -- numero de bits
  PORT   (A, B        : IN  BIT_VECTOR (n-1 DOWNTO 0);     -- entradas do subtrator
          Cin         : IN  BIT;                           -- empresta-1
          S           : OUT BIT_VECTOR (n-1 DOWNTO 0);     -- saida
          Cout        : OUT BIT);                          -- vai-1
END sub_2bits_while;

ARCHITECTURE logica OF sub_2bits_while IS
BEGIN
  PROCESS (A, B, Cin)
    VARIABLE i        : INTEGER ;
    VARIABLE Couti    : BIT_VECTOR (n DOWNTO 0);    -- vai-1 interno
  BEGIN
    i := 0;                                         -- deve ser atualizado a cada iteracao
    Couti(0) := Cin;
    WHILE i <= n-1 LOOP                             -- executado enquanto verdadeiro
      S(i)   <= A(i) XOR B(i)  XOR Couti(i);
      Couti(i+1) := ((not A(i)) AND B(i)) OR ((not A(i)) AND Couti(i)) OR (B(i) AND Couti(i));
      i := i+1;
    END LOOP;
    Cout   <= Couti(n);
  END PROCESS;
END logica;
```

# HDL – Linguagem de Descrição de Hardware

- **Exercício 01:** **Solução para A < B e Cin = 0 → Comando FOR LOOP**

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY sub_2bits_slv IS
  GENERIC    (n       : INTEGER  := 2);                                  -- numero de bits
  PORT    (A, B       : IN  STD_LOGIC_VECTOR (n-1 DOWNTO 0);             -- entradas do subtrator
                Cin    : IN  STD_LOGIC;                                  -- emprest-1
                  S    : OUT STD_LOGIC_VECTOR (n-1 DOWNTO 0);            -- saida
              Cout     : OUT STD_LOGIC;                                  -- vai-1
               Erro    : OUT STD_LOGIC);          -- Erro para A < B  → 1 para resultado negativo
END sub_2bits_slv ;
```

# HDL – Linguagem de Descrição de Hardware

- **Exercício 01: (continuação)** **Solução para A < B e Cin = 0 → Comando FOR LOOP**

```
ARCHITECTURE logica OF sub_2bits_slv IS
BEGIN
            PROCESS (A, B, Cin)
            VARIABLE Couti : STD_LOGIC_VECTOR (n DOWNTO 0);       -- vai-1 interno
            BEGIN
             Couti(0) := Cin;                                     -- Cin = 0 (Forma de Onda)
               IF A < B THEN
                   Erro <= '1';                                   -- Se A < B entao Erro = 1 senao 0
                   S    <= "ZZ";
                   Cout <= 'Z';
               ELSE
                   Erro <= '0';
                   FOR i IN 0 TO n-1 LOOP
                        S(i)   <=  A(i) XOR B(i)  XOR Couti(i);
                        Couti(i+1) := ((not A(i)) AND B(i)) OR ((not A(i)) AND Couti(i)) OR (B(i) AND Couti(i));
                   END LOOP;
                        Cout   <= Couti(n);
                END IF;
            END PROCESS;
END logica;
```

# HDL – Linguagem de Descrição de Hardware

- **Exercício 02:** **Solução para A < B e Cin = 0 → Comando WHILE LOOP**

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY sub_2bits_slv_var IS
  GENERIC    (n      : INTEGER  := 2);                                  -- numero de bits
  PORT    (A, B      : IN  STD_LOGIC_VECTOR (n-1 DOWNTO 0);             -- entradas do subtrator
                Cin   : IN  STD_LOGIC;                                  -- empresta-1
                  S   : OUT STD_LOGIC_VECTOR (n-1 DOWNTO 0);            -- saida
               Cout   : OUT STD_LOGIC;                                  -- vai-1
               Erro   : OUT STD_LOGIC);          -- Erro caso A < B → 1 para resultado negativo
END sub_2bits_slv_var;
```

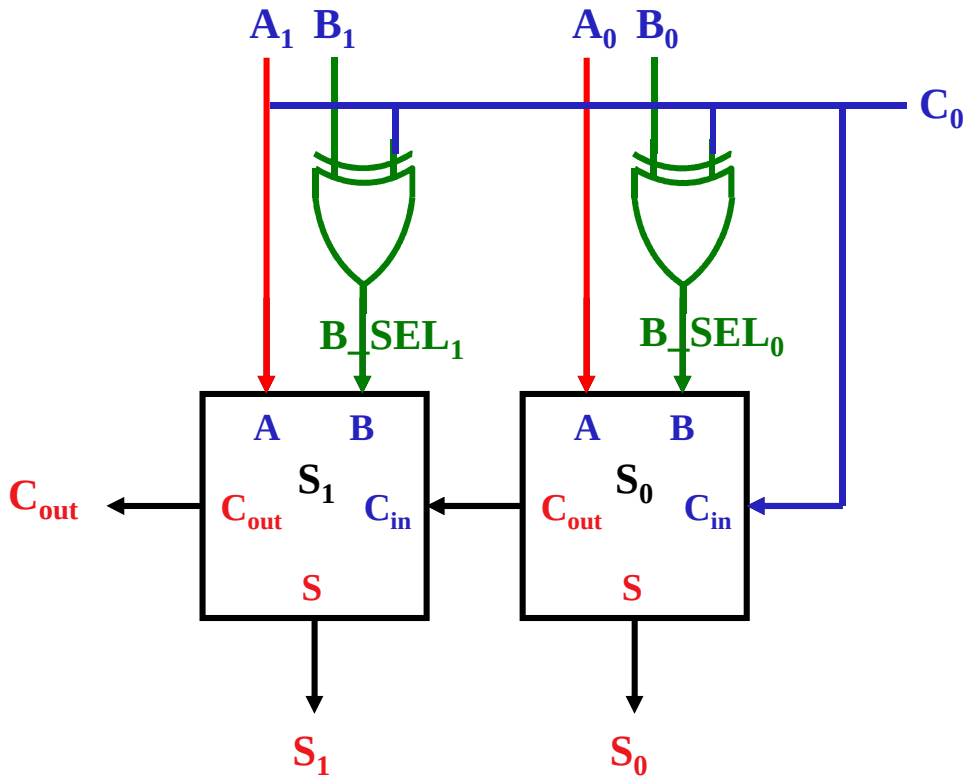
# HDL – Linguagem de Descrição de Hardware

- **Exercício 02: (continuação)** **Solução para A < B e Cin = 0 → Comando WHILE LOOP**

```
ARCHITECTURE logica OF sub_2bits_slv_var IS
BEGIN
          PROCESS (A, B, Cin)
        -- VARIABLE Erro  : STD_LOGIC;                          -- 1 para resultado negativo
          VARIABLE Couti : STD_LOGIC_VECTOR (n DOWNTO 0);       -- vai-1 interno
          BEGIN
           Couti(0) := Cin;                                     -- Cin = 0 (Forma de Onda)
             IF A < B THEN
                Erro <= '1';      -- Erro := '1';       -- Se A < B entao Erro = 1 senao 0
                S    <= "ZZ";
                Cout <= 'Z';
             ELSE
                Erro <= '0';      -- Erro := '0';
                FOR i IN 0 TO n-1 LOOP
                    S(i)   <=  A(i) XOR B(i)  XOR Couti(i);
                    Couti(i+1) := ((not A(i)) AND B(i)) OR ((not A(i)) AND Couti(i)) OR (B(i) AND Couti(i));
                END LOOP;
                Cout   <= Couti(n);
             END IF;
          END PROCESS;
END logica;
```

# HDL - Linguagem de Descrição de Hardware

## Circuito Somador/Subtrator em Complemento de 2




$C_0$=Controle da Operação
$C_0$=0 $\Rightarrow A_i+B_i$
$C_0$=1 $\Rightarrow A_i-B_i$

**Complemento de 2 de B**

$C_0=0$ { $A_0$, $B_0$, $+0$ }  $C_0=1$ { $A_0$, $\overline{B_0}$, $+1$ }

# HDL – Linguagem de Descrição de Hardware

- **Exercício:**

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY som_sub_2bits_C2 IS
  GENERIC (n          : INTEGER  := 2);                          -- numero de bits
  PORT     (A, B      : IN  BIT_VECTOR (n-1 DOWNTO 0);           -- entradas do subtrator
            Cin       : IN  BIT;                                 -- empresta-1
              S       : OUT BIT_VECTOR (n-1 DOWNTO 0);           -- saida
           Cout    : OUT BIT);                                    -- vai-1
END som_sub_2bits_C2 ;
```

# HDL – Linguagem de Descrição de Hardware

- **Exercício:**

```
ARCHITECTURE logica OF som_sub_2bits_C2 IS
BEGIN
   PROCESS (A, B, Cin)
   VARIABLE Couti : BIT_VECTOR (n DOWNTO 0);          -- vai-1 interno
   BEGIN
      IF Cin = '0' THEN
        Couti(0) := Cin;
       FOR i IN 0 TO n-1 LOOP
        S(i) <= A(i) XOR (B(i) XOR Couti(0)) XOR Couti(i);
        Couti(i+1) := (A(i) AND (B(i) XOR Couti(0))) OR (A(i) AND Couti(i)) OR ((B(i) XOR Couti(0)) AND Couti(i));
      END LOOP;
      Cout   <= Couti(n);
      ELSE
        Couti(0) := Cin;
        FOR i IN 0 TO n-1 LOOP
        S(i) <= A(i) XOR (B(i) XOR Couti(0)) XOR Couti(i);
        Couti(i+1) := (A(i) AND (B(i) XOR Couti(0))) OR (A(i) AND Couti(i)) OR ((B(i) XOR Couti(0)) AND Couti(i));
      END LOOP;
      Cout   <= Couti(n);
      END IF;
   END PROCESS;
END logica;
```

# HDL – Linguagem de Descrição de Hardware

- **Exercício:** **Considerando apenas a operação de subtração**

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY sub_2bits_C2 IS
  GENERIC (n          : INTEGER  := 2);                          -- numero de bits
  PORT     (A, B      : IN  STD_LOGIC_VECTOR (n-1 DOWNTO 0);     -- entradas do subtrator
                Cin   : IN  STD_LOGIC;                           -- empresta-1
                  S   : OUT STD_LOGIC_VECTOR (n-1 DOWNTO 0);     -- saida
               Cout   : OUT STD_LOGIC);                          -- vai-1
END sub_2bits_C2 ;
```

# HDL – Linguagem de Descrição de Hardware

- **Exercício:** **Considerando apenas a operação de subtração**

```
ARCHITECTURE logica OF sub_2bits_C2 IS
BEGIN
  PROCESS (A, B, Cin)
  VARIABLE Couti : STD_LOGIC_VECTOR (n DOWNTO 0);          -- vai um interno
  BEGIN
   IF Cin = '1' THEN
    Couti(0) := Cin;
    FOR i IN 0 TO n-1 LOOP
     S(i) <= A(i) XOR (B(i) XOR Couti(0)) XOR Couti(i);
     Couti(i+1) := (A(i) AND (B(i) XOR Couti(0))) OR (A(i) AND Couti(i)) OR ((B(i) XOR Couti(0)) AND Couti(i));
    END LOOP;
    Cout   <= Couti(n);
   ELSE
    S <= "ZZ";
    Cout <= 'Z';
   END IF;
  END PROCESS;
END logica;
```