



**Universidade Federal de Santa Catarina**  
**Centro Tecnológico**  
Departamento de Informática e Estatística  
Curso de Graduação em Ciências da Computação



# **Sistemas Digitais**

**INE 5406**

## **Aula 7-P**

**Descrição em VHDL, síntese e simulação de registradores de deslocamento e contadores. Experimento com a placa DE2 da Altera.**

**Prof. José Luís Güntzel**  
guntzel@inf.ufsc.br

**Est. Vinícius Livramento**  
vini@inf.ufsc.br

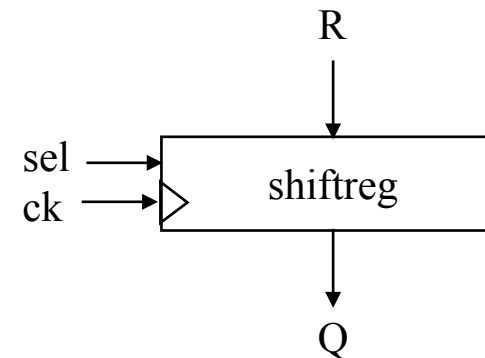
**[www.inf.ufsc.br/~guntzel/ine5406/ine5406.html](http://www.inf.ufsc.br/~guntzel/ine5406/ine5406.html)**

# Registadores de Deslocamento e Contadores

## ▶ Registrador-Deslocador

- Suponha que se deseje descrever em VHDL um registrador capaz de realizar as operações de carga paralela e deslocamento (de 1 bit) para a direita, conforme descrito na tabela de operações abaixo.

sel	Operação	Comentário
0	$Q \leftarrow Q \gg 1$	desl. p/ direita
1	$Q \leftarrow R$	Carrega R

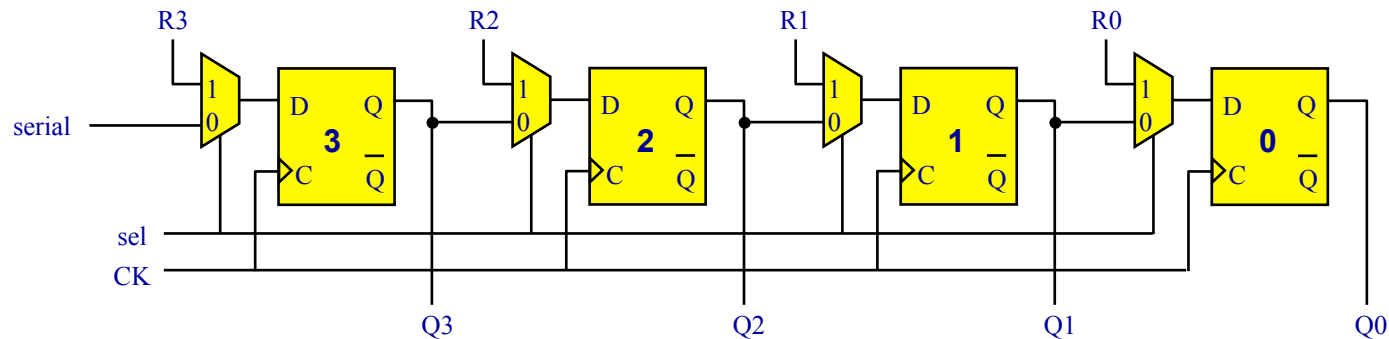


**Observação:** a cada borda de relógio ativa, este registrador realiza uma das duas operações descritas (carga de um valor novo a partir da entrada R ou deslocamento de 1 bit para a direita de seu conteúdo). Portanto, não está prevista a operação “mantém conteúdo”.

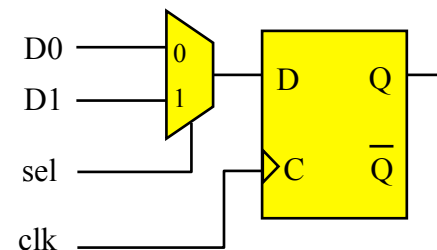
# Registadores de Deslocamento e Contadores

## ▶ Registrador-Deslocador

- O registrador-deslocamento tem a seguinte estrutura.



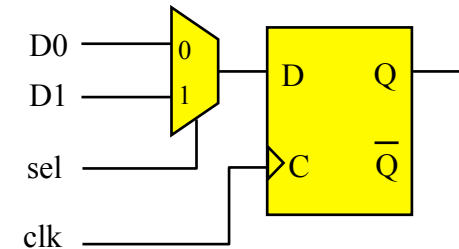
- Uma alternativa para se descrever o registrador-deslocador especificado na transparência anterior é utilizar hierarquia.
- Neste caso, o elemento básico desta estrutura pode ser o seguinte circuito. Chamemo-lo de **muxffd**.



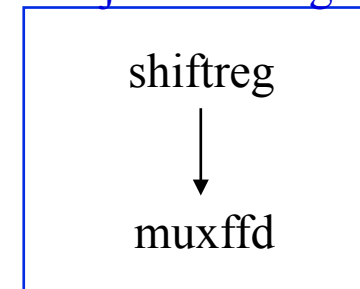
# Registadores de Deslocamento e Contadores

## ▶ Registrador-Deslocador

- Este circuito, por sua vez, poderia ser descrito de **modo hierárquico** (1 mux 2:1 + 1 flip-flop D) ou como **um único** arquivo VHDL (descrição comportamental).
- A segunda descrição é preferível, por ser mais fácil de compreender, ser derivável da descrição convencional de flip-flops e de envolver menos código.
- Hierarquia de arquivos VHDL:



Projeto shiftreg



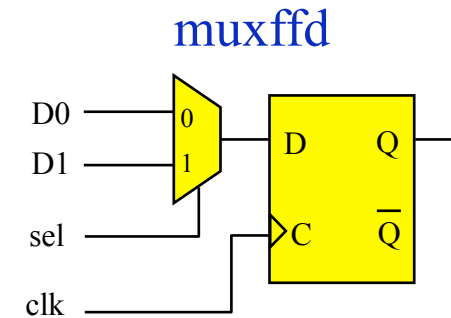
# Registadores de Deslocamento e Contadores

## ▶ Registrador-Deslocador (componente básico)

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

ENTITY muxffd IS
PORT ( D0, D1, sel, clk: IN STD_LOGIC;
      Q : OUT STD_LOGIC );
END muxffd;

ARCHITECTURE comportamento OF muxffd IS
BEGIN
  PROCESS (clk)
  BEGIN
    IF clk'EVENT AND clk = '1' THEN
      IF sel = '0' THEN
        Q <= D0;
      ELSE
        Q <= D1;
      END IF;
    END PROCESS;
  END comportamento;
```



# Registadores de Deslocamento e Contadores

## ▶ Registrador-Deslocador (de 4 bits, hierárquico)

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

```
ENTITY shiftreg IS
```

```
    PORT ( R          : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
          carga, serial, clk : IN STD_LOGIC;
          Q          : BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0) );
```

```
END shiftreg ;
```

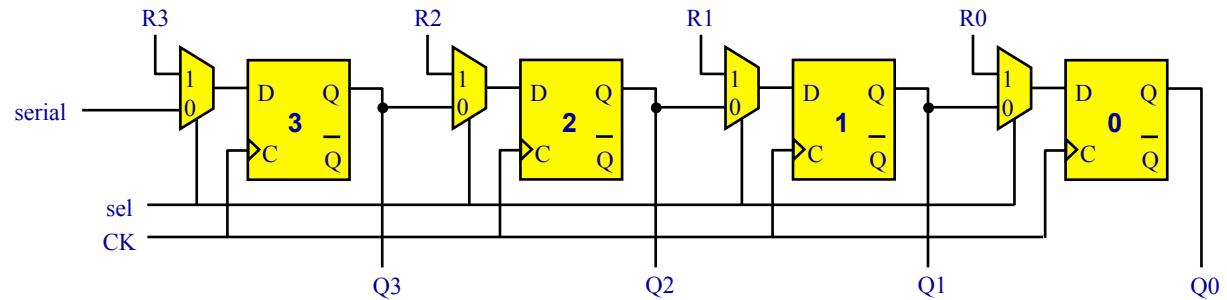
```
ARCHITECTURE estrutura OF shiftreg IS
```

```
    COMPONENT muxffd
        PORT ( D0, D1, sel, clk : IN STD_LOGIC;
              Q                : OUT STD_LOGIC );
    END COMPONENT;
```

```
BEGIN
```

```
    estagio3: muxffd PORT MAP ( serial, R(3), carga, clk, Q(3) );
    estagio2: muxffd PORT MAP ( Q(3), R(2), carga, clk, Q(2) );
    estagio1: muxffd PORT MAP ( Q(2), R(1), carga, clk, Q(1) );
    estagio0: muxffd PORT MAP ( Q(1), R(0), carga, clk, Q(0) );
```

```
END estrutura;
```



→ Por que “BUFFER”?

# Registradores de Deslocamento e Contadores

## ▶ Modos Possíveis para os Sinais que são Portas de Entidade

Modo	Uso
IN	Para sinal que é <b>entrada</b> de entidade
OUT	Para sinal que é <b>saída</b> de entidade. O valor do sinal <b>não pode ser usado dentro da entidade</b> (i.e., o sinal só pode aparecer à esquerda de uma atribuição)
INOUT	Para sinal que é tanto <b>entrada</b> quanto <b>saída</b> de entidade
BUFFER	Para sinal que é <b>saída</b> de entidade. O valor do sinal <b>pode</b> ser usado dentro da entidade (i.e., o sinal pode aparecer tanto à esquerda quanto à direita de uma atribuição)

# Registadores de Deslocamento e Contadores

## ▶ Registrador-Deslocador (descr. comportamental)

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

ENTITY shiftreg2 IS
    PORT ( R          : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
          carga, serial, clk : IN STD_LOGIC;
          Q          : BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0) );
END shiftreg2;
```

```
ARCHITECTURE comportamento OF shiftreg2 IS
BEGIN
```

```
    PROCESS (clk)
```

```
    BEGIN
```

```
        IF clk'EVENT AND clk = '1' THEN
```

```
            IF sel = '1' THEN
```

```
                Q <= R;
```

```
            ELSE
```

```
                Q(0) <= Q(1);
```

```
                Q(1) <= Q(2);
```

```
                Q(2) <= Q(3);
```

```
                Q(3) <= serial;
```

```
            END IF;
```

```
        END PROCESS;
```

```
    END comportamento;
```

sel	Operação	Comentário
0	$Q \leftarrow Q \gg 1$	desl. p/ direita
1	$Q \leftarrow R$	Carrega R



# Registadores de Deslocamento e Contadores

## ▶ Registrador-Deslocador (n bits, descr. comportamental)

ENTITY shiftreg3 IS

**GENERIC ( N : INTEGER := 8 );**

PORT ( R : IN STD\_LOGIC\_VECTOR(N-1 DOWNT0 0);

sel, serial, clk : IN STD\_LOGIC;

Q : BUFFER STD\_LOGIC\_VECTOR(N-1 DOWNT0 0) );

END shiftreg3;

ARCHITECTURE comportamento OF shiftreg3 IS

BEGIN

PROCESS (clk)

BEGIN

IF clk'EVENT AND clk = '1' THEN

IF sel = '1' THEN

Q <= R;

ELSE

**Genbits: FOR i IN 0 TO N-2 LOOP**

**Q(i) <= Q(i+1);**

**END LOOP;**

Q(N-1) <= serial;

END IF;

END IF;

END PROCESS;

END comportamento;

sel	Operação	Comentário
0	$Q \leftarrow Q \gg 1$	desl. p/ direita
1	$Q \leftarrow R$	Carrega R

# Registradores de Deslocamento e Contadores

---

## ▶ FOR LOOP x FOR GENERATE

**FOR LOOP:** usado para gerar um conjunto de atribuições sequenciais (logo, **usado dentro de processos**)

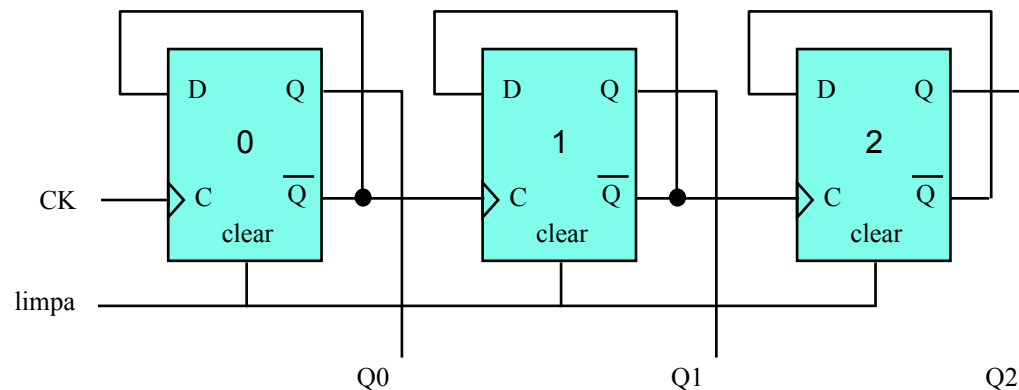
**FOR GENERATE:** usado para gerar um conjunto de atribuições concorrentes (logo, **usado dentro de bloco, mas fora de processos**)

- Ambos declaram variáveis locais que servem apenas para auxiliar (e as vezes, tornar mais genérica) uma descrição VHDL.
- Tratam-se de diretivas que o compilador VHDL compreende e substitui por comportamento/estrutura equivalente em tempo de compilação.

# Registadores de Deslocamento e Contadores

## ▶ Contador de 3 bits (*ripple*)

Usando flip-flops D



- Para descrever este tipo de contador em VHDL usando hierarquia, iremos instanciar o flip-flop D
- Porém, a descrição do flip-flop D em VHDL deve conter ambas saídas, Q e NQ (“not Q”)

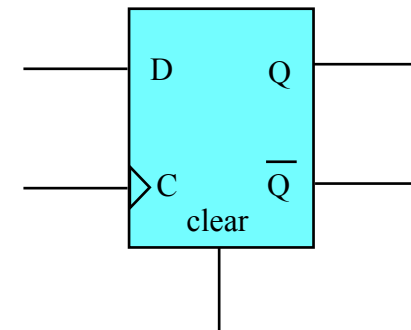
# Registadores de Deslocamento e Contadores

## ▶ Flip-flop D (com saídas Q e NQ)

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

ENTITY ffDv2 IS
PORT ( D, clk, limpa : IN STD_LOGIC;
      Q, NQ : OUT STD_LOGIC);
END ffDv2;

ARCHITECTURE comportamento OF ffDv2 IS
BEGIN
  PROCESS (clk, limpa)
  BEGIN
    IF limpa='1' THEN
      Q <= '0';  NQ <= '1';
    ELSIF clk'EVENT AND clk = '1' THEN
      Q <= D;  NQ <= not D;
    END IF;
  END PROCESS;
END comportamento;
```



# Registadores de Deslocamento e Contadores

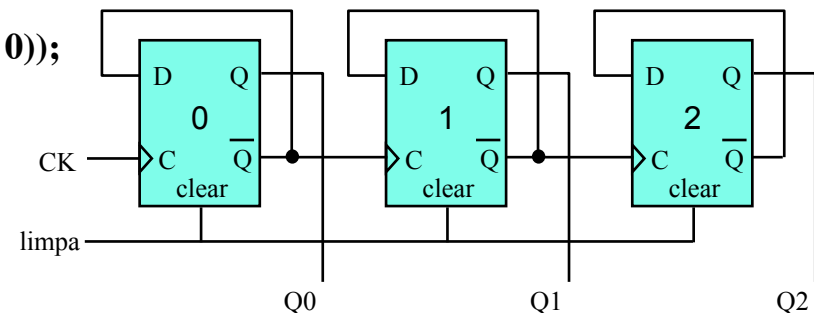
## ▶ Contador de 3 bits (*ripple*)

(Usando flip-flops com saídas Q e NQ)

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
  
ENTITY cont3b IS  
PORT ( clk, limpa : IN STD_LOGIC;  
      Aout : OUT STD_LOGIC_VECTOR (2 DOWNTO 0));  
END cont3b;  
  
ARCHITECTURE estrutura OF cont3b IS  
    SIGNAL NQ0, NQ1, NQ2 : STD_LOGIC;
```

```
    COMPONENT ffDv2  
    PORT ( D, clk, limpa: IN STD_LOGIC; Q, NQ : OUT STD_LOGIC);  
    END COMPONENT;
```

```
BEGIN  
    ffD0: ffDv2 PORT MAP (NQ0, clk, limpa, Aout(0), NQ0);  
    ffD1: ffDv2 PORT MAP (NQ1, NQ0, limpa, Aout(1), NQ1);  
    ffD2: ffDv2 PORT MAP (NQ2, NQ1, limpa, Aout(2), NQ2);  
END estrutura;
```



# Registadores de Deslocamento e Contadores

## ▶ Contador de 4 bits (Versão comportamental 1)

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

ENTITY contador IS
PORT ( clk, limpa , carga : IN    STD_LOGIC;
      Q : OUT  STD_LOGIC_VECTOR( 3 DOWNTO 0) );
END contador ;

ARCHITECTURE comportamento OF contador IS
    SIGNAL conta : STD_LOGIC_VECTOR ( 3 DOWNTO 0) ;
BEGIN
    PROCESS (clk, limpa)
    BEGIN
        IF limpa = '0' THEN
            conta <= "0000";
        ELSIF ( clk'EVENT AND clk = '1' ) THEN
            IF carga = '1' THEN
                conta <= conta + 1;
            ELSE
                conta <= conta; → Não é necessário
                                   (apenas por clareza...)
            END IF;
        END IF;
    END PROCESS;
    Q <= conta;
END comportamento;
```

# Registadores de Deslocamento e Contadores

## ▶ Contador Assíncrono (*ripple*) de n bits

LIBRARY ieee;  
USE ieee.std\_logic\_1164.all; **(Com carga paralela e usando sinais do tipo “integer”)**

ENTITY contnb IS

PORT ( R : IN INTEGER RANGE 0 TO 15;

clk, limpa , carga : IN STD\_LOGIC;

**Q : BUFFER INTEGER RANGE 0 TO 15 );**

END contnb ;

ARCHITECTURE comportamento OF contnb IS

BEGIN

PROCESS (clk, limpa)

BEGIN

IF limpa = '0' THEN

Q <= 0;

ELSIF ( clk'EVENT AND clk = '1' ) THEN

IF carga = '1' THEN

Q <= R;

ELSE

**Q <= Q + 1;**

END IF;

END IF;

END PROCESS;

END comportamento;

limpa	carga	Operação	Comentário
1	0	$Q \leftarrow Q+1$	Conta (incr.conteúdo)
1	1	$Q \leftarrow R$	Carrega R
0	X	$Q \leftarrow 0$	Reseta conteúdo assincronamente

# Projeto de Sistemas Digitais com Ferramentas EDA

---

## ▶ Passos do projeto “contador”

1. Na pasta Meus\_documentos, criar uma pasta com o seu nome (p. ex., “Paulo”). Na pasta “Paulo”, criar uma pasta com nome de “contador”.
2. Acessar o sítio “[www.inf.ufsc.br/~guntzel/ine5406/aula7P](http://www.inf.ufsc.br/~guntzel/ine5406/aula7P)” e baixar para a pasta “contador” os seguintes arquivos:
  - > contador.vhd
  - > reg4bitscont.vhd
  - > binary2dec.vhd
  - > binary2hex.vhd
  - > Setup\_Cyclone\_2C35\_DE2.tcl
3. Abrir o Quartus II e criar um projeto na pasta “contador”, usando “contador.vhd” como toplevel. Escolher o dispositivo FPGA EP2C35F672C6 e selecionar o ModelSim-Altera como EDA Simulation Tool.
4. Compilar o projeto criado.



# Prototipação com a Placa DE2 da Altera

## ▶ Verificar o mapeamento dos pinos do FPGA

The screenshot shows the Quartus II Pin Planner interface. The 'Pins' menu is open, showing options like 'Device...', 'Timing Analysis Settings...', 'EDA Tool Settings...', 'Settings...', 'Classic Timing Analyzer Wizard...', 'Assignment Editor', 'Pin Planner', 'Remove Assignments...', 'Demote Assignments...', 'Back-Annotate Assignments...', 'Import Assignments...', 'Export Assignments...', 'Assignment (Time) Groups...', 'Timing Closure Floorplan', 'LogicLock Regions Window', and 'Design Partitions Window'. The 'Top View - Wire Bond' window displays the Cyclone II EP2C35F672C7 pin grid. A table at the bottom lists pin assignments for nodes HEX0[6:0].

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Group
HEX0[6]	Output			3.3-V LVTTTL (default)			HEX0[6..0]
HEX0[5]	Output			3.3-V LVTTTL (default)			HEX0[6..0]
HEX0[4]	Output			3.3-V LVTTTL (default)			HEX0[6..0]
HEX0[3]	Output			3.3-V LVTTTL (default)			HEX0[6..0]
HEX0[2]	Output			3.3-V LVTTTL (default)			HEX0[6..0]
HEX0[1]	Output			3.3-V LVTTTL (default)			HEX0[6..0]
HEX0[0]	Output			3.3-V LVTTTL (default)			HEX0[6..0]

# Prototipação com a Placa DE2 da Altera

## ▶ Verificar o mapeamento dos pinos do FPGA

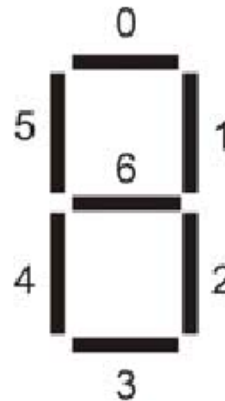
Mapeamentos do switches

Abrir o “DE2 UserManual.pdf”

Mapeamentos dos displays de 7 segmentos

Signal Name	FPGA Pin No.	Description
SW[0]	PIN_N25	Toggle Switch[0]
SW[1]	PIN_N26	Toggle Switch[1]
SW[2]	PIN_P25	Toggle Switch[2]
SW[3]	PIN_AE14	Toggle Switch[3]
SW[4]	PIN_AF14	Toggle Switch[4]
SW[5]	PIN_AD13	Toggle Switch[5]
SW[6]	PIN_AC13	Toggle Switch[6]
SW[7]	PIN_C13	Toggle Switch[7]
SW[8]	PIN_B13	Toggle Switch[8]
SW[9]	PIN_A13	Toggle Switch[9]
SW[10]	PIN_N1	Toggle Switch[10]
SW[11]	PIN_P1	Toggle Switch[11]
SW[12]	PIN_P2	Toggle Switch[12]
SW[13]	PIN_T7	Toggle Switch[13]
SW[14]	PIN_U3	Toggle Switch[14]
SW[15]	PIN_U4	Toggle Switch[15]
SW[16]	PIN_V1	Toggle Switch[16]
SW[17]	PIN_V2	Toggle Switch[17]

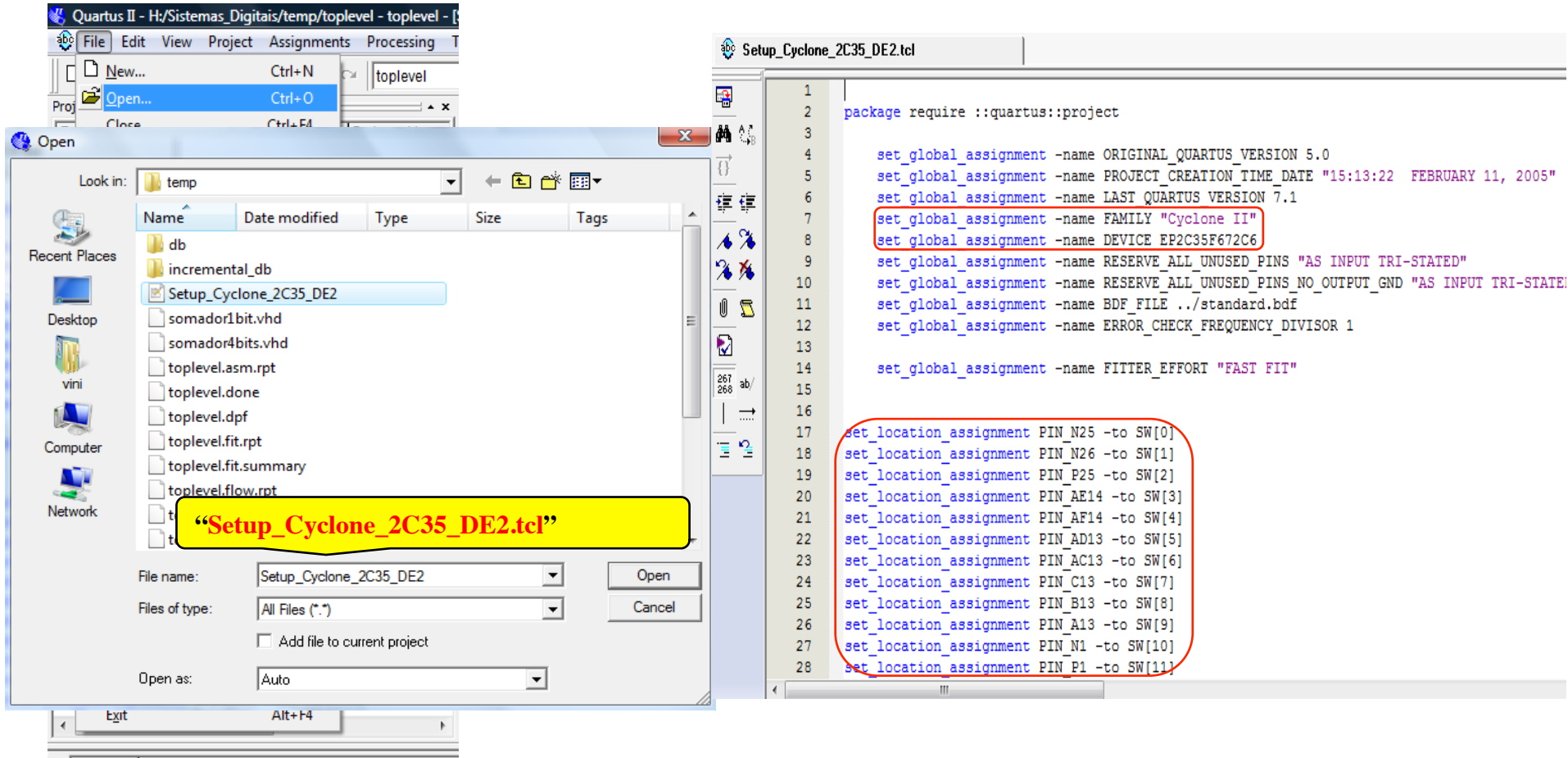
Posição e index de cada segmento do display de 7 segmentos



Signal Name	FPGA Pin No.	Description
HEX0[0]	PIN_AF10	Seven Segment Digit 0[0]
HEX0[1]	PIN_AB12	Seven Segment Digit 0[1]
HEX0[2]	PIN_AC12	Seven Segment Digit 0[2]
HEX0[3]	PIN_AD11	Seven Segment Digit 0[3]
HEX0[4]	PIN_AE11	Seven Segment Digit 0[4]
HEX0[5]	PIN_V14	Seven Segment Digit 0[5]
HEX0[6]	PIN_V13	Seven Segment Digit 0[6]
HEX1[0]	PIN_V20	Seven Segment Digit 1[0]
HEX1[1]	PIN_V21	Seven Segment Digit 1[1]
HEX1[2]	PIN_W21	Seven Segment Digit 1[2]
HEX1[3]	PIN_Y22	Seven Segment Digit 1[3]
HEX1[4]	PIN_AA24	Seven Segment Digit 1[4]
HEX1[5]	PIN_AA23	Seven Segment Digit 1[5]
HEX1[6]	PIN_AB24	Seven Segment Digit 1[6]
HEX2[0]	PIN_AB23	Seven Segment Digit 2[0]
HEX2[1]	PIN_V22	Seven Segment Digit 2[1]

# Prototipação com a Placa DE2 da Altera

## ▶ O script “Setup\_Cyclone\_2C35\_DE2.tcl”

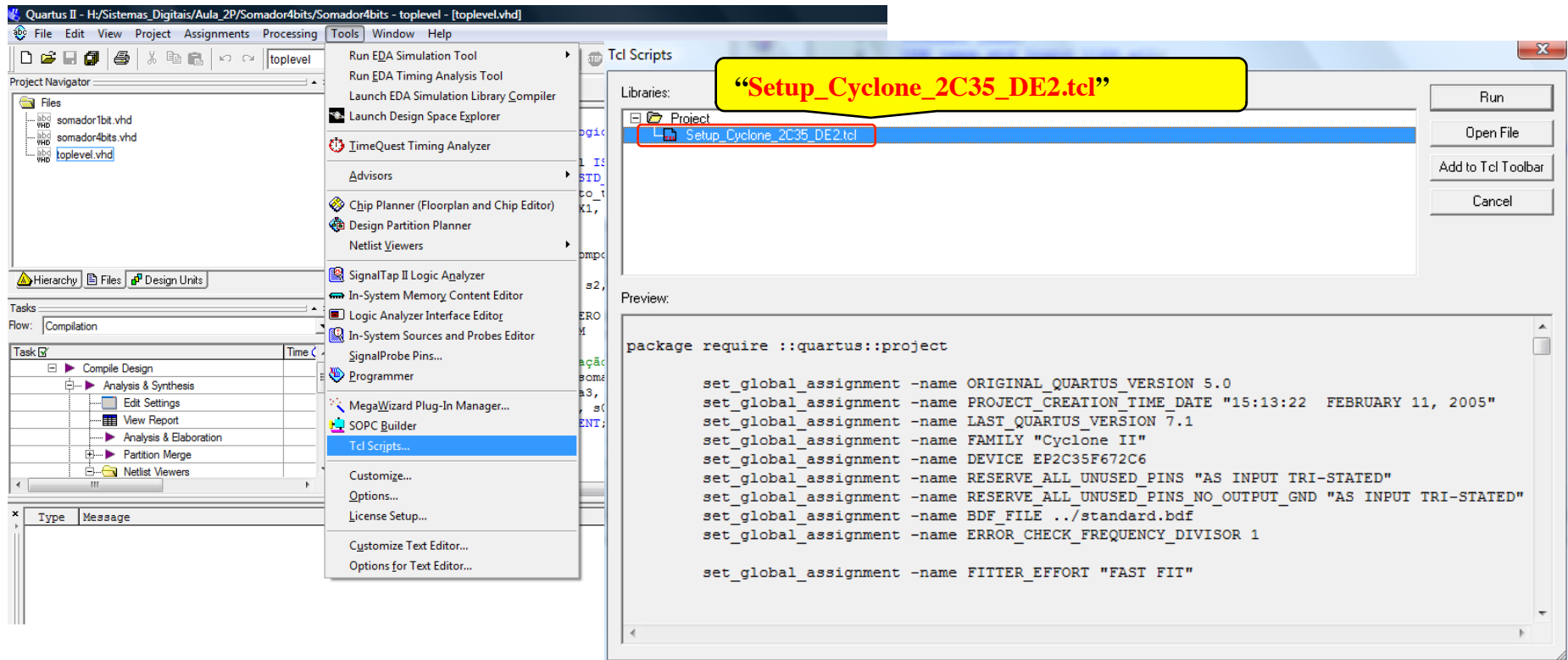


The image shows a screenshot of the Quartus II software interface. On the left, an 'Open' dialog box is displayed, showing the file 'Setup\_Cyclone\_2C35\_DE2' selected in the 'temp' directory. The file name is entered in the 'File name' field, and the 'Files of type' is set to 'All Files (\*.\*)'. A yellow callout box highlights the file name 'Setup\_Cyclone\_2C35\_DE2.tcl'. On the right, the 'Setup\_Cyclone\_2C35\_DE2.tcl' script is shown in a text editor. The script contains several global and location assignments. Two lines are highlighted with red boxes: 'set\_global\_assignment -name FAMILY "Cyclone II"' and 'set\_global\_assignment -name DEVICE EP2C35F672C6'. A larger red box highlights a block of location assignments from line 17 to 28, including 'set\_location\_assignment PIN\_N25 -to SW[0]', 'set\_location\_assignment PIN\_N26 -to SW[1]', 'set\_location\_assignment PIN\_P25 -to SW[2]', 'set\_location\_assignment PIN\_AE14 -to SW[3]', 'set\_location\_assignment PIN\_AF14 -to SW[4]', 'set\_location\_assignment PIN\_AD13 -to SW[5]', 'set\_location\_assignment PIN\_AC13 -to SW[6]', 'set\_location\_assignment PIN\_C13 -to SW[7]', 'set\_location\_assignment PIN\_B13 -to SW[8]', 'set\_location\_assignment PIN\_A13 -to SW[9]', 'set\_location\_assignment PIN\_N1 -to SW[10]', and 'set\_location\_assignment PIN\_P1 -to SW[11]'.

```
1 package require ::quartus::project
2
3
4 set_global_assignment -name ORIGINAL_QUARTUS_VERSION 5.0
5 set_global_assignment -name PROJECT_CREATION_TIME_DATE "15:13:22 FEBRUARY 11, 2005"
6 set_global_assignment -name LAST_QUARTUS_VERSION 7.1
7 set_global_assignment -name FAMILY "Cyclone II"
8 set_global_assignment -name DEVICE EP2C35F672C6
9
10 set_global_assignment -name RESERVE_ALL_UNUSED_PINS "AS INPUT TRI-STATE"
11 set_global_assignment -name RESERVE_ALL_UNUSED_PINS_NO_OUTPUT_GND "AS INPUT TRI-STATE"
12 set_global_assignment -name BDF_FILE ../standard.bdf
13 set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 1
14
15 set_global_assignment -name FITTER_EFFORT "FAST FIT"
16
17 set_location_assignment PIN_N25 -to SW[0]
18 set_location_assignment PIN_N26 -to SW[1]
19 set_location_assignment PIN_P25 -to SW[2]
20 set_location_assignment PIN_AE14 -to SW[3]
21 set_location_assignment PIN_AF14 -to SW[4]
22 set_location_assignment PIN_AD13 -to SW[5]
23 set_location_assignment PIN_AC13 -to SW[6]
24 set_location_assignment PIN_C13 -to SW[7]
25 set_location_assignment PIN_B13 -to SW[8]
26 set_location_assignment PIN_A13 -to SW[9]
27 set_location_assignment PIN_N1 -to SW[10]
28 set_location_assignment PIN_P1 -to SW[11]
```

# Prototipação com a Placa DE2 da Altera

## ▶ Executar script de mapeamento dos pinos



# Prototipação com a Placa DE2 da Altera

## ▶ Verificar o mapeamento dos pinos do FPGA

The screenshot shows the Quartus II Pin Planner interface. The top view of the Cyclone II EP2C35F672C6 FPGA is displayed, showing the pin locations and their assignments. A table below the top view lists the pin assignments for nodes SW[0] through SW[6].

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Group
SW[0]	Unknown	PIN_N25	5	B5_N1	3.3-V LVTTTL (default)		
SW[1]	Unknown	PIN_N26	5	B5_N1	3.3-V LVTTTL (default)		
SW[2]	Unknown	PIN_P25	6	B6_N0	3.3-V LVTTTL (default)		
SW[3]	Unknown	PIN_AE14	7	B7_N1	3.3-V LVTTTL (default)		
SW[4]	Unknown	PIN_AF14	7	B7_N1	3.3-V LVTTTL (default)		
SW[5]	Unknown	PIN_AD13	8	B8_N0	3.3-V LVTTTL (default)		
SW[6]	Unknown	PIN_AC13	8	B8_N0	3.3-V LVTTTL (default)		

# Prototipação com a Placa DE2 da Altera

---

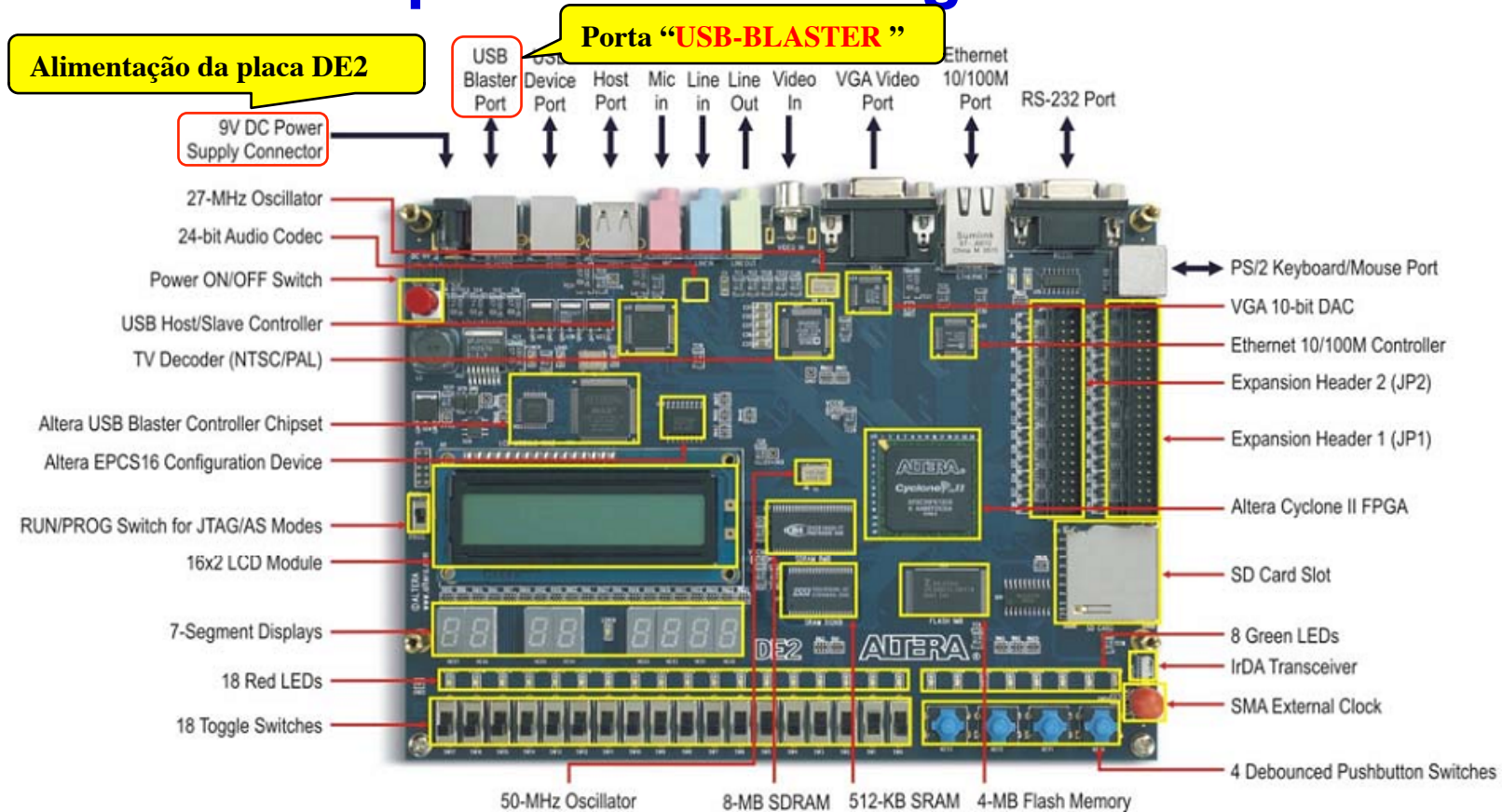
## ▶ Passos do projeto “contador”

### Compilar

1. Processing -> Start Compilation
2. Aguardar mensagem “Quartus II Full Compilation Succesfull” (ou mensagem de erro)

# Prototipação com a Placa DE2 da Altera

## ▶ Conectar a placa no PC e ligá-la



# Prototipação com a Placa DE2 da Altera

## ▶ Baixar o projeto para a placa

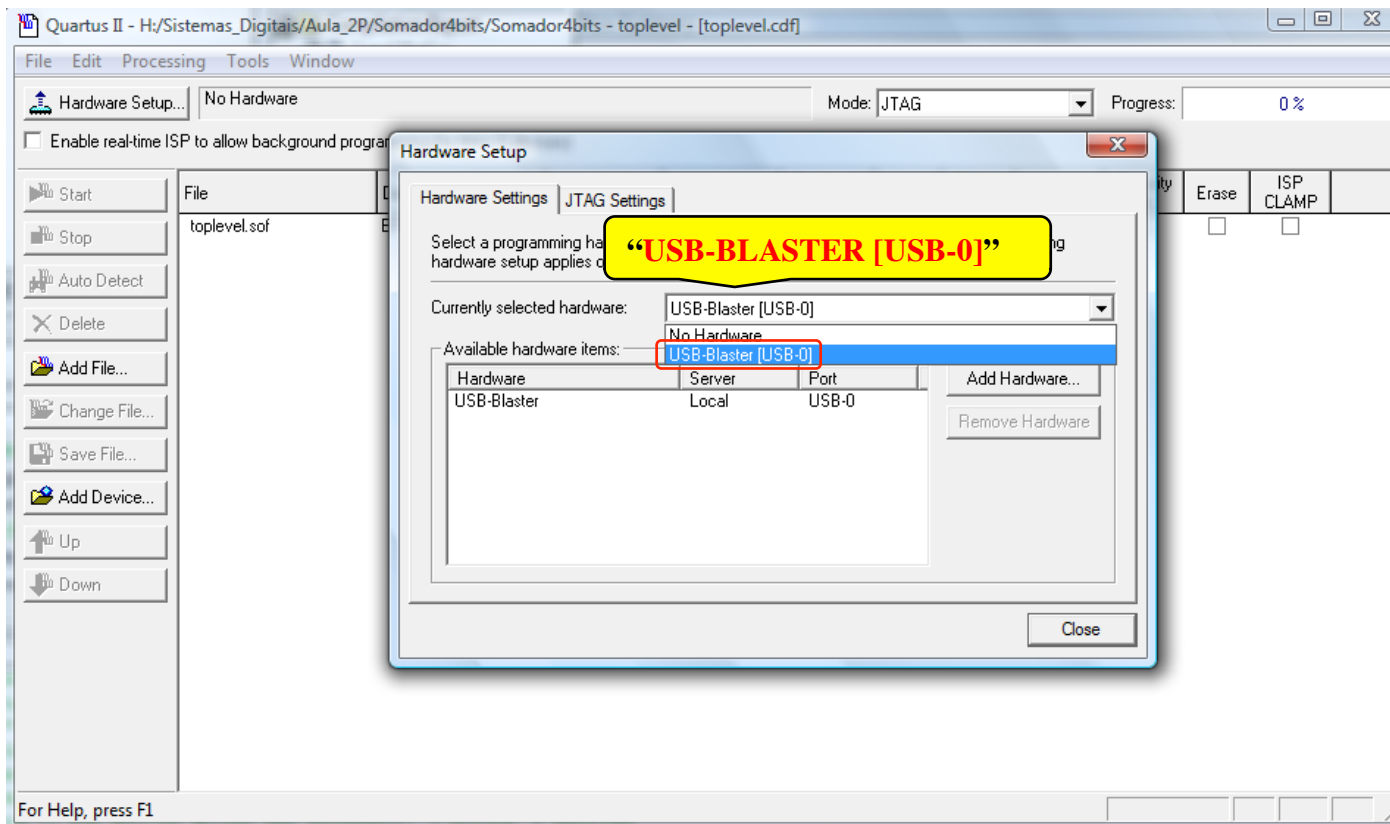
The screenshot displays the Quartus II IDE interface. The 'Tools' menu is open, and the 'Programmer' option is highlighted. The background shows the Project Navigator with files like 'somador1bit.vhd', 'somador4bits.vhd', and 'toplevel.vhd'. The Tasks window shows the compilation flow, and the Compilation Report - Flow Summary window displays the following data:

Flow Status	Success
Quartus II Version	9.1 Build
Revision Name	toplevel
Top-level Entity Name	toplevel
Family	Cyclone
Device	EP2K10K10
Timing Models	Final
Met timing requirements	Yes
Total logic elements	10 / 33,2
Total combinational functions	10 / 33,2
Dedicated logic registers	0 / 33,2
Total registers	0
Total pins	77 / 475
Total virtual pins	0
Total memory bits	0 / 483,2
Embedded Multiplier 9-bit elements	0 / 70 (0)
Total PLLs	0 / 4 (0)



# Prototipação com a Placa DE2 da Altera

## ▶ Baixar o projeto para a placa



# Prototipação com a Placa DE2 da Altera

## ▶ Baixar o projeto para a placa

