Laboratório 3 - Modelação através de estadogramas e implementação em VHDL

Objetivos:

Pretende-se modelar (e implementar, na aula ou fora dela) um relógio com horas, minutos e segundos, dispondo de um visualizador com 4 dígitos. Considere que dispõe de um evento que é gerado a um ritmo de 20ns (com base no sinal de relógio disponível nos kits de trabalho, obtido do oscilador de 50 MHz).
Com base no relógio obtido, pretende-se modelar e implementar um relógio com horas, minutos e segundos, dispondo de um visualizador com 4 dígitos e que integre as funções de despertador, cronómetro e/ou temporizador.
A solução hardware será obtida através da codificação direta em VHDL a partir dos modelos comportamentais dos vários componentes e o seu funcionamento deverá ser verificado no kit Spartan-3.

Estrutura do enunciado:

Este enunciado está dividido em três partes, em que a primeira foca as questões de modelação, enquanto a segunda se prende com questões de implementação e a terceira com funcionalidades adicionais (que poderão maioritariamente ser alvo de trabalho fora da aula).

Parte I – Modelação de um relógio com estadogramas

a) Apresente um estadograma para modelar esse relógio. Compare duas atitudes de modelação distintas;
   - a primeira, mais abstracta, identificando os vários subsistemas e suas interdependências (isto é, a componente de horas, a de minutos, a de segundos,...), e
   - a segunda, mais concreta, baseando-se nos recursos a utilizar (isto é, em componentes de modelação directamente associadas à implementação).

   Tomando a componente para modelar os minutos, enquanto na primeira atitude temos um modelo para descrever o estado de contagem dos minutos (de 0 a 59), na segunda temos dois modelos para descrever os estados de contagem das dezenas e unidades de minuto. Assim, na primeira atitude, as saídas para actuação nos *displays* são obtidas por descodificação do estado de contagem (conversor de código binário natural para dois dígitos BCD), na segunda, as variáveis contendo os estados de contagem podem ser interligadas directamente aos módulos de display.

b) Considere que dispõe de 2 botões, um de acerto para horas e outro para minutos (como nos relógios disponíveis nalguns automóveis); amplie o modelo obtido na alínea a) de modo a integrar estas novas funcionalidades. Desta forma, o nosso modelo/sistema está a ser desenvolvido de forma incremental.

c) Considere que substitui os dois botões da alínea anterior por outros dois, com funções de "selecção de modo" e "incremento", respectivamente (como nos relógios disponíveis em alguns equipamentos de microondas). Altere o modelo obtido na alínea a) de modo a integrar estas novas funcionalidades. Compare duas atitudes de modelação distintas, a primeira baseada na utilização dos sinais associados aos botões e a segunda baseada nos eventos a eles associados.

Parte II – Implementação utilizando VHDL e o ambiente ISE/WebPack da Xilinx

Pretende-se utilizar o ambiente de especificação e simulação ISE da Xilinx.

Pretende-se que:

1- Codifique em VHDL o modelo obtido na Parte I, recorrendo à codificação de contadores. A título de exemplo, apresenta-se nas páginas seguintes o código associado à codificação de um relógio de minutos e segundos. Importa chamar a atenção para uma parte do código apresentado, dado representar uma regra de bom desenho. Em particular quando se pretende que o contador de minutos seja incrementado quando se recebe o sinal *min_u* dependente do andar a montante da unidade de segundos, bem como do flanco do sinal de relógio, a especificação utilizada é:

```
elsif clk'event and clk = '1' then
  if min_u = '1' then
    ...
  end if;
end if;
```

e não a seguinte (aparentemente equivalente do ponto de vista lógico):

```
elsif clk'event and clk = '1' and min_u = '1' then
  ...
end if;
```

uma vez que a arquitetura inferida pela ferramenta de síntese a partir da primeira especificação é um elemento de memória cujo sinal de relógio é *clk* e em que o sinal *min_u* (entre outros) irá determinar o estado das entradas síncronas do elemento de memória; na segunda especificação a ferramenta de síntese irá inferir um elemento de memória em que o sinal de relógio será gerado por uma função lógica em que se inclui a condição em *min_u*, não sendo utilizada a linha de distribuição de relógio disponível do dispositivo, com as consequências associadas.

2- Complete a codificação do relógio analisado no ponto anterior, de forma a incluir as horas, bem como as funcionalidade previstas nas partes I b) e c).

3- Relativamente ao código utilizado como exemplo para um relógio de minutos+segundos, implemente variantes para visualização de horas+minutos, bem como uma outra que visualizará horas+minutos ou minutos+segundos de acordo com o estado de um botão de entrada adicional; ou ainda variantes em que o código do relógio não disponibiliza os códigos dos dígitos em BCD, mas o código binário natural associado.

4- proceda à simulação do sistema;

5- proceda à configuração da FPGA e verifique o funcionamento no circuito.

Especificação *VHDL* relativa ao modelo estadograma do relógio

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity relogio is
    Port ( clk : in std_logic;
           reset : in std_logic;
           c_seg_u : out integer range 0 to 9;
           c_seg_d : out integer range 0 to 5;
           c_min_u : out integer range 0 to 9;
           c_min_d : out integer range 0 to 5);
end relogio;
```

```
architecture Behavioral of relogio is

signal cont_subseg : integer range 0 to 50000000;
signal cont_seg_u, cont_min_u : integer range 0 to 9;
signal cont_seg_d, cont_min_d : integer range 0 to 5;
signal subseg, seg_u, seg_d, min_u, min_d : std_logic;

begin

process (clk, reset)
begin
 if reset = '1' then
    cont_subseg <= 0;
  elsif clk'event and clk = '1' then
    if cont_subseg = 49999999 then
      cont_subseg <= 0;
    else cont_subseg <= cont_subseg + 1;
    end if;
  end if;
end process;

seg_u <= '1' when cont_subseg = 49999999 else '0';

process (clk, reset)
begin
  if reset = '1' then
    cont_seg_u <= 0;
  elsif clk'event and clk = '1' then
    if seg_u = '1' then
      if cont_seg_u = 9 then
        cont_seg_u <= 0;
      else cont_seg_u <= cont_seg_u + 1;
      end if;
    end if;
  end if;
end process;

seg_d <= '1' when cont_seg_u = 9 and seg_u = '1' else '0';
c_seg_u <= cont_seg_u;

process (clk, reset)
begin
  if reset = '1' then
    cont_seg_d <= 0;
  elsif clk'event and clk = '1' then
    if seg_d = '1' then
      if cont_seg_d = 5 then
        cont_seg_d <= 0;
      else cont_seg_d <= cont_seg_d + 1;
      end if;
    end if;
  end if;
end process;

min_u <= '1' when cont_seg_d = 5 and seg_d = '1' else '0';
c_seg_d <= cont_seg_d;

process (clk, reset)
begin
  if reset = '1' then
```

```
    cont_min_u <= 0;
  elsif clk'event and clk = '1' then
    if min_u = '1' then
      if cont_min_u = 9 then
        cont_min_u <= 0;
      else cont_min_u <= cont_min_u + 1;
      end if;
    end if;
  end if;
end process;

min_d <= '1' when cont_min_u = 9 and min_u = '1' else '0';
c_min_u <= cont_min_u;

process (clk, reset)
begin
  if reset = '1' then
    cont_min_d <= 0;
  elsif clk'event and clk = '1' then
    if min_d = '1' then
      if cont_min_d = 5 then
        cont_min_d <= 0;
      else cont_min_d <= cont_min_d + 1;
      end if;
    end if;
  end if;
end process;

horas_u <= '1' when cont_min_d = 9 and min_d = '1' else '0';
c_min_d <= cont_min_d;

end Behavioral;
```

Especificação *VHDL* relativa ao conversor de código para displays

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity conv_displays is
    Port ( clk : in std_logic;
           reset : in std_logic;
           seg_u : in integer range 0 to 9;
           seg_d : in integer range 0 to 5;
           min_u : in integer range 0 to 9;
           min_d : in integer range 0 to 5;
           an3 : out std_logic;
           an2 : out std_logic;
           an1 : out std_logic;
           an0 : out std_logic;
           algarismo : out std_logic_vector(7 downto 0));
end conv_displays;

architecture Behavioral of conv_displays is

signal alg : integer range 0 to 9;
signal cont_clock : integer range 0 to 499999;
signal num_alg : std_logic_vector(1 downto 0);
```

```
begin

co: process (clk, reset)
begin
 if reset = '1' then
    cont_clock <= 0;
  elsif clk'event and clk = '1' then
    if cont_clock = 499999 then
      cont_clock <= 0;
      num_alg <= "00";
    else
      if cont_clock = 124999 then
        num_alg <= "01";
      elsif cont_clock = 249999 then
        num_alg <= "10";
      elsif cont_clock = 374999 then
        num_alg <= "11";
      end if;
      cont_clock <= cont_clock + 1;
    end if;
  end if;
end process;

an : process(clk, reset)
begin
  if reset = '1' then
    an3 <= '0';
    an2 <= '0';
    an1 <= '0';
    an0 <= '0';
    alg <= 0;
    algarismo(7) <= '1';
  elsif clk'event and clk = '1' then
    if num_alg = "00" then
      an3 <= '1';
      an2 <= '1';
      an1 <= '1';
      an0 <= '0';
      alg <= seg_u;
      algarismo(7) <= '1';
    elsif num_alg = "01" then
      an3 <= '1';
      an2 <= '1';
      an1 <= '0';
      an0 <= '1';
      alg <= seg_d;
      algarismo(7) <= '1';
    elsif num_alg = "10" then
      an3 <= '1';
      an2 <= '0';
      an1 <= '1';
      an0 <= '1';
      alg <= min_u;
      algarismo(7) <= '0';
    elsif num_alg = "11" then
      an3 <= '0';
      an2 <= '1';
      an1 <= '1';
      an0 <= '1';
      alg <= min_d;
```

```
      algarismo(7) <= '1';
    end if;
  end if;
end process;

al : process(clk, reset)
begin
  if reset = '1' then
      algarismo(6 downto 0) <= "0000001";
  elsif clk'event and clk = '1' then
    case alg is
      when 0 => algarismo(6 downto 0) <= "0000001";
      when 1 => algarismo(6 downto 0) <= "1001111";
      when 2 => algarismo(6 downto 0) <= "0010010";
      when 3 => algarismo(6 downto 0) <= "0000110";
      when 4 => algarismo(6 downto 0) <= "1001100";
      when 5 => algarismo(6 downto 0) <= "0100100";
      when 6 => algarismo(6 downto 0) <= "0100000";
      when 7 => algarismo(6 downto 0) <= "0001111";
      when 8 => algarismo(6 downto 0) <= "0000000";
      when 9 => algarismo(6 downto 0) <= "0000100";
      when others => algarismo(6 downto 0) <= "0111000";
    end case;
   end if;
end process;

end Behavioral;
```

Especificação VHDL relativa à interligação dos vários módulos

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity principal is
    Port ( reset : in std_logic;
           clk : in std_logic;
           algarismo : out std_logic_vector(7 downto 0);
           an3 : out std_logic;
           an2 : out std_logic;
           an1 : out std_logic;
           an0 : out std_logic);
end principal;

architecture Behavioral of principal is

      COMPONENT relogio
      PORT(
            clk : IN std_logic;
            reset : IN std_logic;
            c_seg_u : OUT std_logic_vector(0 to 3);
            c_seg_d : OUT std_logic_vector(0 to 2);
            c_min_u : OUT std_logic_vector(0 to 3);
            c_min_d : OUT std_logic_vector(0 to 2)
            );
      END COMPONENT;

      COMPONENT conv_displays
      PORT(
```

```
            clk : IN std_logic;
            reset : IN std_logic;
            seg_u : IN std_logic_vector(0 to 3);
            seg_d : IN std_logic_vector(0 to 2);
            min_u : IN std_logic_vector(0 to 3);
            min_d : IN std_logic_vector(0 to 2);
            an3 : OUT std_logic;
            an2 : OUT std_logic;
            an1 : OUT std_logic;
            an0 : OUT std_logic;
            algarismo : OUT std_logic_vector(7 downto 0)
            );
      END COMPONENT;

signal c_seg_u_aux, c_min_u_aux : std_logic_vector(0 to 3);
signal c_seg_d_aux, c_min_d_aux : std_logic_vector(0 to 2);

begin

  r : relogio
  port map(reset => reset,
           clk => clk,
           c_seg_u => c_seg_u_aux,
           c_seg_d => c_seg_d_aux,
           c_min_u => c_min_u_aux,
           c_min_d => c_min_d_aux
           );

  cd : conv_displays
  port map(reset => reset,
           clk => clk,
           seg_u => c_seg_u_aux,
           seg_d => c_seg_d_aux,
           min_u => c_min_u_aux,
           min_d => c_min_d_aux,
           algarismo => algarismo,
           an3 => an3,
           an2 => an2,
           an1 => an1,
           an0 => an0
           );

end Behavioral;
```

Conteúdo do ficheiro UCF associado para atribuição dos pinos

```
NET "algarismo<0>"  LOC = "N16"  ;
NET "algarismo<1>"  LOC = "F13"  ;
NET "algarismo<2>"  LOC = "R16"  ;
NET "algarismo<3>"  LOC = "P15"  ;
NET "algarismo<4>"  LOC = "N15"  ;
NET "algarismo<5>"  LOC = "G13"  ;
NET "algarismo<6>"  LOC = "E14"  ;
NET "algarismo<7>"  LOC = "P16"  ;
NET "an0"  LOC = "D14"  ;
NET "an1"  LOC = "G14"  ;
NET "an2"  LOC = "F14"  ;
NET "an3"  LOC = "E13"  ;
NET "clk"  LOC = "T9"  ;
NET "reset"  LOC = "L14"  ;
```

Parte III – Modelação e implementação de funcionalidades adicionais

Partindo do relógio realizado na Parte II do enunciado, pretende-se que vá integrando novas funcionalidades ao sistema, na sequência abaixo referida.
Tenha presente que para algumas alíneas necessitará de considerar uma decomposição do sistema em termos de parte de controlo e de parte de dados, isto é, enquanto alguns objetivos estão associados à parte de controlo (modelada através de um estadograma), outros objetivos provocam alterações na parte de dados (por exemplo, seleção do que deve ser visualizado, se horas+minutos, se minutos+segundos).

Sequência proposta:

a) Implementação completa do relógio de horas, minutos e segundos, com visualização de horas+minutos ou minutos+segundos, dependendo de uma variável de controlo externa. Necessário decidir entre a utilização de um interruptor (permitindo que a opção se verifique em permanência) ou de um botão de pressão (permitindo que a opção seja ativa apenas enquanto de pressiona o botão). Tendo em conta a opção desejada, necessita de alterar o ficheiro UCF associado.

b) Adição ao estadograma da alínea anterior de novas componentes permitindo obter um despertador/alarme, isto é adicionando contadores de horas e minutos de despertador/alarme. A parte de dados será responsável por gerar o sinal de alarme (despertador) através da comparação da hora corrente com a hora de alarme. A função será ativada/desativada através de um interruptor de entrada. Considere que o alarme se manterá ativo durante o máximo de uma hora ou até o interruptor de alarme ativo for desativado.
Complementarmente, adicionar mecanismos de acerto quer de hora corrente, quer de hora de alarme. Uma solução vulgarmente utilizada considera três sinais de entrada:
- o primeiro permitindo selecionar o que se pode alterar, se hora corrente, se hora de alarme;
- o segundo permitindo uma acerto lento (tipicamente variando os minutos ao ritmo do segundo);
- o terceiro permitindo um acerto rápido (tipicamente dez vezes mais rápido).

c) Adição ao sistema de uma função de temporizador para controlo de uma tomada (que permite ligar uma saída durante um tempo pré-definido nunca superior a 59 minutos, útil para, por exemplo, controlar uma lâmpada ou outro equipamento elétrico à noite). Considere uma entrada para activar/desactivar a funcionalidade, bem como, opcionalmente, um mecanismo de acerto do intervalo de tempo.

d) Adição ao sistema de uma função de cronómetro, isto é, um bloco que possa ser inicializado a zero, e posteriormente disparado e parado permitindo a contagem de segundos+centésimos de segundo.

Nota: Considerando a diversidade de modelos disponíveis, admite-se que variantes ao enunciado proposto serão “naturais”.