Laboratório Digital I - PCS3635

Planejamento da Experiência 3:

Desenvolvendo um Fluxo de Dados

Marco Aurélio C. O. Prado - NUSP 11257605
Victor Hoefling Padula - NUSP 10770051
Turma 04 - Bancada A1

São Paulo - SP
20/01/2021

1) Objetivo

Os objetivos da aula consistem em aprender sobre:

- Projeto de circuitos usando descrição estrutural VHDL;
- Projeto de uma unidade de controle;
- Documentação de projetos (planejamento e relatório);
- Estudo de um fluxo de dados de um circuito digital.

2) Elaboração do Circuito “circuito_exp4.vhd”

2.1) Descrição VHDL do circuito “fluxo_dados.vhd”

```
----------------------------------------------------------------
-- Arquivo : contador_163.vhd
-- Projeto : Experiencia 01 - Primeiro Contato com VHDL
----------------------------------------------------------------
-- Descricao : contador binario hexadecimal (modulo 16)
-- similar ao CI 74163
----------------------------------------------------------------
-- Revisoes :
-- Data Versao Autor Descricao
-- 29/12/2020 1.0 Edson Midorikawa criacao
----------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity contador_163 is -- entidade principal
 port (
clock : in std_logic; -- sinais de entrada
clr : in std_logic;
ld : in std_logic;
ent : in std_logic;
enp : in std_logic;
D : in std_logic_vector (3 downto 0);
Q : out std_logic_vector (3 downto 0); -- sinais de saída
rco : out std_logic
 );
end contador_163;
architecture comportamental of contador_163 is -- declaração da
arquitetura
 signal IQ: integer range 0 to 15;
begin
```

```
 process (clock,ent,IQ) -- inicio do process do circuito
 begin
if clock'event and clock='1' then
-- as mudanças no circuito ocorrem com o clock em 1
if clr='0' then IQ <= 0;
-- caso o sinal clear seja 0, a contagem é reiniciada
elsif ld='0' then IQ <= to_integer(unsigned(D));
-- caso o sinal load seja 0, a entrada D é carregada
elsif ent='1' and enp='1' then
-- ambos os sinais de controle precisam estar em 1
-- para que a contagem seja realizada
if IQ=15 then IQ <= 0;
-- caso chegue no final da contagem, volta p/ 0
else IQ <= IQ + 1;
-- caso contrário, soma-se 1 no contador
end if;
else IQ <= IQ;
-- caso um dos dois sinais de controle não esteja em nível
-- lógico alto, o contador permanece em seu estado atual
end if;
end if;
if IQ=15 and ent='1' then rco <= '1';
-- caso o contador tenha chegado no final, rco assume valor 1
else rco <= '0';
end if;
Q <= std_logic_vector(to_unsigned(IQ, Q'length));
-- a saída Q recebe o valor do sinal utilizado para a contagem
 end process; -- fim do process
end comportamental; -- fim da arquitetura




----------------------------------------------------------------------
-- Arquivo   : comparador_85.vhd
-- Projeto   : Experiencia 02 - Um Fluxo de Dados Simples
----------------------------------------------------------------------
-- Descricao : comparador binario de 4 bits
--           similar ao CI 7485
--           baseado em descricao criada por Edson Gomi (11/2017)
```

```
-------------------------------------------------------------------
-- Revisoes  :
--  Data      Versao  Autor           Descricao
--  02/01/2021  1.0   Edson Midorikawa  criacao
-------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity comparador_85 is  -- declaracao da entidade do comparador
  port (  -- entradas
  i_A3   : in  std_logic;
  i_B3   : in  std_logic;
  i_A2   : in  std_logic;
  i_B2   : in  std_logic;
  i_A1   : in  std_logic;
  i_B1   : in  std_logic;
  i_A0   : in  std_logic;
  i_B0   : in  std_logic;
  i_AGTB : in  std_logic;
  i_ALTB : in  std_logic;
  i_AEQB : in  std_logic;
  -- saidas
  o_AGTB : out std_logic;
  o_ALTB : out std_logic;
  o_AEQB : out std_logic
  );
end entity comparador_85;

architecture dataflow of comparador_85 is -- inicio da arquitetura
do comparador
  -- sinais intermediarios que comparam A e B sem levar em
consideracao as entradas de cascateamento
 signal agtb : std_logic;
  signal aeqb : std_logic;
  signal altb : std_logic;
begin
  -- equacoes dos sinais: pagina 462, capitulo 6 do livro-texto
  -- Wakerly, J.F. Digital Design - Principles and Practice, 4th
Edition
```

```
  -- veja tambem datasheet do CI SN7485 (Function Table)
  agtb <= (i_A3 and not(i_B3)) or
        (not(i_A3 xor i_B3) and i_A2 and not(i_B2)) or
        (not(i_A3 xor i_B3) and not(i_A2 xor i_B2) and i_A1 and
not(i_B1)) or
        (not(i_A3 xor i_B3) and not(i_A2 xor i_B2) and not(i_A1 xor
i_B1) and i_A0 and not(i_B0));
  -- checa se a > b
  aeqb <= not((i_A3 xor i_B3) or (i_A2 xor i_B2) or (i_A1 xor i_B1)
or (i_A0 xor i_B0));
  -- checa se a = b
  altb <= not(agtb or aeqb);
  -- checa se a < b
  o_AGTB <= agtb or (aeqb and (not(i_AEQB) and not(i_ALTB)));
  o_ALTB <= altb or (aeqb and (not(i_AEQB) and not(i_AGTB)));
  o_AEQB <= aeqb and i_AEQB;
  -- nas saidas, são levadas em consideracao as entradas de
cascateamento para obter um resultado final

end architecture dataflow; -- fim da arquitetura

-------------------------------------------------------------------
-- Arquivo    : ram_16x4.vhd
-- Projeto    : Experiencia 03 - Desenvolvendo o Fluxo de Dados
-------------------------------------------------------------------
-- Descricao : módulo de memória RAM 16x4
--             sinais we e ce ativos em baixo
--             codigo ADAPTADO do código encontrado no livro
--               VHDL Descricao e Sintese de Circuitos Digitais
--               Roberto D'Amore, LTC Editora.
-------------------------------------------------------------------
-- Revisoes  :
--     Data        Versao  Autor             Descricao
--     08/01/2020  1.0     Edson Midorikawa  criacao
-------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
```

```
entity ram_16x4 is
   port (
       endereco     : in  std_logic_vector(3 downto 0);
       dado_entrada : in  std_logic_vector(3 downto 0);
       we           : in  std_logic;
       ce           : in  std_logic;
       dado_saida   : out std_logic_vector(3 downto 0)
    );
end entity ram_16x4;

architecture ram1 of ram_16x4 is
  type   arranjo_memoria is array(0 to 15) of std_logic_vector(3
downto 0);
  signal memoria : arranjo_memoria;
  attribute ram_init_file: string;
  attribute ram_init_file of memoria: signal is
"ram_conteudo_jogadas.mif";
begin

  process(ce, we, endereco, memoria)
  begin
    if ce = '0' then -- dado armazenado na subida de "we" com "ce=0"
      if rising_edge(we)
          then memoria(to_integer(unsigned(endereco))) <=
dado_entrada;
      end if;
    end if;
    dado_saida <= memoria(to_integer(unsigned(endereco)));
  end process;

end architecture ram1;









library ieee;
```

```vhdl
use ieee.std_logic_1164.all;

entity fluxo_dados is
    port (
        clock : in std_logic;

        zeraC : in std_logic;

        contaC : in std_logic;
        escreveM: in std_logic;
        chaves : in std_logic_vector (3 downto 0);
        fimC: out std_logic;
        db_contagem : out std_logic_vector (3 downto 0);
        db_memoria: out std_logic_vector(3 downto 0);


        igual: out std_logic
    );
end entity fluxo_dados;

architecture estrutural of fluxo_dados is

    component contador_163 is
        port (
        clock : in  std_logic;
        clr   : in  std_logic;
        ld    : in  std_logic;
        ent   : in  std_logic;
        enp   : in  std_logic;
        D     : in  std_logic_vector (3 downto 0);
        Q     : out std_logic_vector (3 downto 0);
        rco   : out std_logic
   );
    end component;



    component comparador_85 is
        port (  -- entradas
  i_A3   : in  std_logic;
```

```
  i_B3   : in  std_logic;
  i_A2   : in  std_logic;
  i_B2   : in  std_logic;
  i_A1   : in  std_logic;
  i_B1   : in  std_logic;
  i_A0   : in  std_logic;
  i_B0   : in  std_logic;
  i_AGTB : in  std_logic;
  i_ALTB : in  std_logic;
  i_AEQB : in  std_logic;
  -- saidas
  o_AGTB : out std_logic;
  o_ALTB : out std_logic;
  o_AEQB : out std_logic
        );
    end component;




    component ram_16x4 is
        port(
          endereco: in std_logic_vector(3 downto 0);
          dado_entrada: in std_logic_vector(3 downto 0);
          we: in std_logic;
          ce: in std_logic;
          dado_saida: out std_logic_vector(3 downto 0)
        );
    end component;




    signal
great,ZeraC_baixo,igual_out,menor,great_o,menor_o,enable_cin,rco_out
c: std_logic;
    signal contador_out, s_dado,s_endereco: std_logic_vector (3
downto 0);
    begin
```

```
    ZeraC_baixo<= not ZeraC;
  Contend: contador_163 port map (
      clock => clock,
      clr => ZeraC_baixo,
      ld    => '1',
      ent   => enable_Cin,
      enp   =>enable_Cin,
      D     => "0000",
      Q     => s_endereco,
      rco   => rco_outC
  );
enable_Cin<=contaC;
db_contagem<=s_endereco;
  compJog: comparador_85 port map (
      i_A3  =>s_dado(3),
      i_B3   => chaves(3),
      i_A2   =>s_dado(2),
      i_B2   => chaves(2),
      i_A1   =>s_dado(1),
      i_B1   => chaves(1),
      i_A0   =>s_dado(0),
      i_B0   => chaves(0),
      i_AGTB =>'0',
      i_ALTB => '0',
      i_AEQB => '1',
      -- saidas
      o_AGTB =>great,
      o_ALTB =>menor,
      o_AEQB => igual_out
  );


  memoria: ram_16x4 port map(
      endereco => s_endereco,
      dado_entrada => chaves,
      we => escreveM,
      ce => '0',
      dado_saida => s_dado
```

```
        );
        db_memoria<=s_dado;
     igual<=igual_out;
    fimC<=rco_outC;
    end architecture;
```

2.2) Descrição VHDL do circuito “unidade_controle.vhd”

```
-----------------------------------------------------------------
-- Arquivo    : unidade_controle.vhd
-- Projeto    : Experiencia 04 - Projeto de uma Unidade de Controle
-----------------------------------------------------------------
-- Descricao : codificação da máquina de estados em VHDL
--             > para mais informação, consulte o livro
--               VHDL Descricao e Sintese de Circuitos Digitais
--               Roberto D'Amore, LTC Editora.
--               ou acesse a postagem do prof, Bruno Albertini em
--               https://balbertini.github.io/vhdl_fsm-pt_BR.html
-----------------------------------------------------------------
-- Revisoes  :
--     Data        Versao  Autor             Descricao
--     21/01/2020  1.0     Edson Midorikawa  criacao
-----------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity unidade_controle is
  port (
    clock:     in  std_logic;
    reset:     in  std_logic;
    iniciar:   in  std_logic;
    fim:       in  std_logic;
    zera:      out std_logic;
    conta:     out std_logic;
    pronto:    out std_logic;
    db_estado: out std_logic_vector(3 downto 0)
  );
```

```
end entity;

architecture fsm of unidade_controle is
  type t_estado is (A, B, C, D, E);
  signal Eatual, Eprox: t_estado;
begin
  -- memoria de estado
  process (clock,reset)
  begin
    if reset='1' then
      Eatual <= A;
    elsif clock'event and clock = '1' then
      Eatual <= Eprox;
    end if;
  end process;

  -- logica de proximo estado
  Eprox <=
      A when  Eatual=A and iniciar='0' else
      B when  Eatual=A and iniciar='1' else
      C when  Eatual=B else
      D when  Eatual=C and fim='0' else
      E when  Eatual=C and fim='1' else
      C when  Eatual=D else
      A when  Eatual=E else
      A;

  -- logica de saída (maquina de Moore)
  with Eatual select
    zera <=   '0' when A | C | D | E,
              '1' when B,
              '0' when others;

  with Eatual select
    conta <=  '0' when A | B | C | E,
              '1' when D,
              '0' when others;

  with Eatual select
    pronto <= '0' when A | B | C | D,
```

```
              '1' when E,
              '0' when others;

  -- saida de depuracao (db_estado)
  with Eatual select
    db_estado <= "0000" when A,      -- 0
                 "1011" when B,      -- B
                 "1100" when C,      -- C
                 "1101" when D,      -- D
                 "1110" when E,      -- E
                 "1111" when others; -- F
end fsm;
```

2.3) Descrição VHDL do circuito “circuito_exp4.vhd”

```
-----------------------------------------------------------------
-- Arquivo    : unidade_controle.vhd
-- Projeto    : Experiencia 04 - Projeto de uma Unidade de Controle
-----------------------------------------------------------------
-- Descricao : codificação da máquina de estados em VHDL
--             > para mais informação, consulte o livro
--               VHDL Descricao e Sintese de Circuitos Digitais
--               Roberto D'Amore, LTC Editora.
--               ou acesse a postagem do prof, Bruno Albertini em
--               https://balbertini.github.io/vhdl_fsm-pt_BR.html
-----------------------------------------------------------------
-- Revisoes  :
--     Data        Versao  Autor             Descricao
--     21/01/2020  1.0     Edson Midorikawa  criacao
-----------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity unidade_controle is
  port (
    clock:     in  std_logic;
    reset:     in  std_logic;
    iniciar:   in  std_logic;
    fim:       in  std_logic;
    zera:      out std_logic;
```

```
    conta:     out std_logic;
    pronto:    out std_logic;
    db_estado: out std_logic_vector(3 downto 0)
  );
end entity;

architecture fsm of unidade_controle is
  type t_estado is (A, B, C, D, E);
  signal Eatual, Eprox: t_estado;
begin
  -- memoria de estado
  process (clock,reset)
  begin
    if reset='1' then
      Eatual <= A;
    elsif clock'event and clock = '1' then
      Eatual <= Eprox;
    end if;
  end process;

  -- logica de proximo estado
  Eprox <=
      A when  Eatual=A and iniciar='0' else
      B when  Eatual=A and iniciar='1' else
      C when  Eatual=B else
      D when  Eatual=C and fim='0' else
      E when  Eatual=C and fim='1' else
      C when  Eatual=D else
      A when  Eatual=E else
      A;

  -- logica de saída (maquina de Moore)
  with Eatual select
    zera <=   '0' when A | C | D | E,
              '1' when B,
              '0' when others;

  with Eatual select
    conta <=  '0' when A | B | C | E,
              '1' when D,
```

```
              '0' when others;

  with Eatual select
    pronto <= '0' when A | B | C | D,
              '1' when E,
              '0' when others;

  -- saida de depuracao (db_estado)
  with Eatual select
    db_estado <= "0000" when A,      -- 0
                 "1011" when B,      -- B
                 "1100" when C,      -- C
                 "1101" when D,      -- D
                 "1110" when E,      -- E
                 "1111" when others; -- F
end fsm;


-------------------------------------------------------------------
-- Arquivo : contador_163.vhd
-- Projeto : Experiencia 01 - Primeiro Contato com VHDL
-------------------------------------------------------------------
-- Descricao : contador binario hexadecimal (modulo 16)
-- similar ao CI 74163
-------------------------------------------------------------------
-- Revisoes :
-- Data Versao Autor Descricao
-- 29/12/2020 1.0 Edson Midorikawa criacao
-------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity contador_163 is -- entidade principal
 port (
clock : in std_logic; -- sinais de entrada
clr : in std_logic;
ld : in std_logic;
ent : in std_logic;
enp : in std_logic;
D : in std_logic_vector (3 downto 0);
```

```
Q : out std_logic_vector (3 downto 0); -- sinais de saída
rco : out std_logic
 );
end contador_163;
architecture comportamental of contador_163 is -- declaração da
arquitetura
 signal IQ: integer range 0 to 15;
begin
 process (clock,ent,IQ) -- inicio do process do circuito
 begin
if clock'event and clock='1' then
-- as mudanças no circuito ocorrem com o clock em 1
if clr='0' then IQ <= 0;
-- caso o sinal clear seja 0, a contagem é reiniciada
elsif ld='0' then IQ <= to_integer(unsigned(D));
-- caso o sinal load seja 0, a entrada D é carregada
elsif ent='1' and enp='1' then
-- ambos os sinais de controle precisam estar em 1
-- para que a contagem seja realizada
if IQ=15 then IQ <= 0;
-- caso chegue no final da contagem, volta p/ 0
else IQ <= IQ + 1;
-- caso contrário, soma-se 1 no contador
end if;
else IQ <= IQ;
-- caso um dos dois sinais de controle não esteja em nível
-- lógico alto, o contador permanece em seu estado atual
end if;
end if;
if IQ=15 and ent='1' then rco <= '1';
-- caso o contador tenha chegado no final, rco assume valor 1
else rco <= '0';
end if;
Q <= std_logic_vector(to_unsigned(IQ, Q'length));
-- a saída Q recebe o valor do sinal utilizado para a contagem
 end process; -- fim do process
end comportamental; -- fim da arquitetura
```

```
-------------------------------------------------------------------
-- Arquivo    : comparador_85.vhd
-- Projeto    : Experiencia 02 - Um Fluxo de Dados Simples
-------------------------------------------------------------------
-- Descricao : comparador binario de 4 bits
--           similar ao CI 7485
--           baseado em descricao criada por Edson Gomi (11/2017)
-------------------------------------------------------------------
-- Revisoes  :
--  Data        Versao  Autor            Descricao
--  02/01/2021  1.0     Edson Midorikawa  criacao
-------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity comparador_85 is  -- declaracao da entidade do comparador
  port (  -- entradas
    i_A3   : in  std_logic;
    i_B3   : in  std_logic;
    i_A2   : in  std_logic;
    i_B2   : in  std_logic;
    i_A1   : in  std_logic;
    i_B1   : in  std_logic;
    i_A0   : in  std_logic;
    i_B0   : in  std_logic;
    i_AGTB : in  std_logic;
    i_ALTB : in  std_logic;
    i_AEQB : in  std_logic;
    -- saidas
    o_AGTB : out std_logic;
    o_ALTB : out std_logic;
    o_AEQB : out std_logic
  );
end entity comparador_85;

architecture dataflow of comparador_85 is -- inicio da arquitetura
do comparador
```

```
  -- sinais intermediarios que comparam A e B sem levar em
consideracao as entradas de cascateamento
 signal agtb : std_logic;
  signal aeqb : std_logic;
  signal altb : std_logic;
begin
  -- equacoes dos sinais: pagina 462, capitulo 6 do livro-texto
  -- Wakerly, J.F. Digital Design - Principles and Practice, 4th
Edition
  -- veja tambem datasheet do CI SN7485 (Function Table)
  agtb <= (i_A3 and not(i_B3)) or
        (not(i_A3 xor i_B3) and i_A2 and not(i_B2)) or
        (not(i_A3 xor i_B3) and not(i_A2 xor i_B2) and i_A1 and
not(i_B1)) or
        (not(i_A3 xor i_B3) and not(i_A2 xor i_B2) and not(i_A1 xor
i_B1) and i_A0 and not(i_B0));
  -- checa se a > b
  aeqb <= not((i_A3 xor i_B3) or (i_A2 xor i_B2) or (i_A1 xor i_B1)
or (i_A0 xor i_B0));
  -- checa se a = b
  altb <= not(agtb or aeqb);
  -- checa se a < b
  o_AGTB <= agtb or (aeqb and (not(i_AEQB) and not(i_ALTB)));
  o_ALTB <= altb or (aeqb and (not(i_AEQB) and not(i_AGTB)));
  o_AEQB <= aeqb and i_AEQB;
  -- nas saidas, são levadas em consideracao as entradas de
cascateamento para obter um resultado final

end architecture dataflow; -- fim da arquitetura


-------------------------------------------------------------------
-- Arquivo    : ram_16x4.vhd
-- Projeto    : Experiencia 03 - Desenvolvendo o Fluxo de Dados
-------------------------------------------------------------------
-- Descricao : módulo de memória RAM 16x4
--             sinais we e ce ativos em baixo
--             codigo ADAPTADO do código encontrado no livro
--               VHDL Descricao e Sintese de Circuitos Digitais
--               Roberto D'Amore, LTC Editora.
-------------------------------------------------------------------
```

```
-- Revisoes  :
--     Data        Versao  Autor             Descricao
--     08/01/2020  1.0     Edson Midorikawa  criacao
-----------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ram_16x4 is
   port (
       endereco     : in  std_logic_vector(3 downto 0);
       dado_entrada : in  std_logic_vector(3 downto 0);
       we           : in  std_logic;
       ce           : in  std_logic;
       dado_saida   : out std_logic_vector(3 downto 0)
    );
end entity ram_16x4;

architecture ram1 of ram_16x4 is
  type   arranjo_memoria is array(0 to 15) of std_logic_vector(3
downto 0);
  signal memoria : arranjo_memoria;
  attribute ram_init_file: string;
  attribute ram_init_file of memoria: signal is
"ram_conteudo_jogadas.mif";
begin

  process(ce, we, endereco, memoria)
  begin
    if ce = '0' then -- dado armazenado na subida de "we" com "ce=0"
      if rising_edge(we)
          then memoria(to_integer(unsigned(endereco))) <=
dado_entrada;
      end if;
    end if;
    dado_saida <= memoria(to_integer(unsigned(endereco)));
  end process;

end architecture ram1;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity fluxo_dados is
    port (
        clock : in std_logic;

        zeraC : in std_logic;

        contaC : in std_logic;
        escreveM: in std_logic;
        chaves : in std_logic_vector (3 downto 0);
        fimC: out std_logic;
        db_contagem : out std_logic_vector (3 downto 0);
        db_memoria: out std_logic_vector(3 downto 0);


        igual: out std_logic
    );
end entity fluxo_dados;

architecture estrutural of fluxo_dados is

    component contador_163 is
        port (
        clock : in  std_logic;
        clr   : in  std_logic;
        ld    : in  std_logic;
        ent   : in  std_logic;
        enp   : in  std_logic;
        D     : in  std_logic_vector (3 downto 0);
        Q     : out std_logic_vector (3 downto 0);
```

```
        rco   : out std_logic
    );
    end component;



    component comparador_85 is
        port (  -- entradas
    i_A3   : in  std_logic;
    i_B3   : in  std_logic;
    i_A2   : in  std_logic;
    i_B2   : in  std_logic;
    i_A1   : in  std_logic;
    i_B1   : in  std_logic;
    i_A0   : in  std_logic;
    i_B0   : in  std_logic;
    i_AGTB : in  std_logic;
    i_ALTB : in  std_logic;
    i_AEQB : in  std_logic;
    -- saidas
    o_AGTB : out std_logic;
    o_ALTB : out std_logic;
    o_AEQB : out std_logic
        );
    end component;



    component ram_16x4 is
        port(
          endereco: in std_logic_vector(3 downto 0);
          dado_entrada: in std_logic_vector(3 downto 0);
          we: in std_logic;
          ce: in std_logic;
          dado_saida: out std_logic_vector(3 downto 0)
        );
    end component;
```

```
    signal
great,ZeraC_baixo,igual_out,menor,great_o,menor_o,enable_cin,rco_out
c: std_logic;
    signal contador_out, s_dado,s_endereco: std_logic_vector (3
downto 0);
    begin


          ZeraC_baixo<= not ZeraC;
        Contend: contador_163 port map (
            clock => clock,
            clr => ZeraC_baixo,
            ld    => '1',
            ent   => enable_Cin,
            enp   =>enable_Cin,
            D     => "0000",
            Q     => s_endereco,
            rco   => rco_outC
        );
      enable_Cin<=contaC;
      db_contagem<=s_endereco;
        compJog: comparador_85 port map (
            i_A3  =>s_dado(3),
              i_B3   => chaves(3),
              i_A2   =>s_dado(2),
              i_B2   => chaves(2),
              i_A1   =>s_dado(1),
              i_B1   => chaves(1),
              i_A0   =>s_dado(0),
              i_B0   => chaves(0),
              i_AGTB =>'0',
              i_ALTB => '0',
              i_AEQB => '1',
              -- saidas
              o_AGTB =>great,
              o_ALTB =>menor,
              o_AEQB => igual_out
```

```
        );


        memoria: ram_16x4 port map(
            endereco => s_endereco,
            dado_entrada => chaves,
            we => escreveM,
            ce => '0',
            dado_saida => s_dado
        );
        db_memoria<=s_dado;
     igual<=igual_out;
    fimC<=rco_outC;
    end architecture;

-------------------------------------------------------------------
-- Arquivo    : hexa7seg.vhd
-- Projeto    : Jogo do Desafio da Memoria
-------------------------------------------------------------------
-- Descricao : decodificador hexadecimal para
--             display de 7 segmentos
--
-- entrada: hexa - codigo binario de 4 bits hexadecimal
-- saida:   sseg - codigo de 7 bits para display de 7 segmentos
-------------------------------------------------------------------
-- dica de uso: mapeamento para displays da placa DE0-CV
--              bit 6 mais significativo é o bit a esquerda
--              p.ex. sseg(6) -> HEX0[6] ou HEX06
-------------------------------------------------------------------
-- Revisoes  :
--     Data        Versao  Autor             Descricao
--     29/12/2020  1.0     Edson Midorikawa  criacao
-------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity hexa7seg is
    port (
        hexa : in  std_logic_vector(3 downto 0);
```

```
        sseg : out std_logic_vector(6 downto 0)
    );
end hexa7seg;

architecture comportamental of hexa7seg is
begin

  sseg <= "1000000" when hexa="0000" else
          "1111001" when hexa="0001" else
          "0100100" when hexa="0010" else
          "0110000" when hexa="0011" else
          "0011001" when hexa="0100" else
          "0010010" when hexa="0101" else
          "0000010" when hexa="0110" else
          "1111000" when hexa="0111" else
          "0000000" when hexa="1000" else
          "0000000" when hexa="1000" else
          "0010000" when hexa="1001" else
          "0001000" when hexa="1010" else
          "0000011" when hexa="1011" else
          "1000110" when hexa="1100" else
          "0100001" when hexa="1101" else
          "0000110" when hexa="1110" else
          "0001110" when hexa="1111" else
          "1111111";

end comportamental;




library ieee;
use ieee.std_logic_1164.all;
entity circuito_exp4 is
    port(
        clock : in std_logic;
        reset : in std_logic;
        iniciar : in std_logic;
        chaves : in std_logic_vector (3 downto 0);
```

```
        pronto : out std_logic;
        db_igual : out std_logic;
        db_contagem : out std_logic_vector (6 downto 0);
        db_memoria : out std_logic_vector (6 downto 0);
        db_estado : out std_logic_vector (6 downto 0)
    );
end entity;

architecture estrutural of circuito_exp4 is
    component fluxo_dados is
        port(
            clock : in std_logic;

            zeraC : in std_logic;

            contaC : in std_logic;
            escreveM: in std_logic;
            chaves : in std_logic_vector (3 downto 0);
            fimC: out std_logic;
            db_contagem : out std_logic_vector (3 downto 0);
            db_memoria: out std_logic_vector(3 downto 0);


            igual: out std_logic
        );
    end component;

    component unidade_controle is
        port(
            clock:     in  std_logic;
            reset:     in  std_logic;
            iniciar:   in  std_logic;
            fim:       in  std_logic;
            zera:      out std_logic;
            conta:     out std_logic;
            pronto:    out std_logic;
            db_estado: out std_logic_vector(3 downto 0)
        );
    end component;
```

```
component hexa7seg is
    port (
        hexa : in  std_logic_vector(3 downto 0);
        sseg : out std_logic_vector(6 downto 0)
    );
    end component;

signal conta4,memo4,estad4: std_logic_vector (3 downto 0);
signal conta,fim,zera: std_logic;
begin
    FD: fluxo_dados port map(
        clock => clock,

        zeraC => zera,

        contaC => conta,
        escreveM=> '1',
        chaves => chaves,
        fimC=> fim,
        db_contagem => conta4,
        db_memoria=> memo4,
        igual=>db_igual
    );

    UC: unidade_controle port map(
        clock=>clock,
        reset=>reset,
        iniciar=>iniciar,
        fim=>fim,
        zera=> zera,
        conta=> conta,
        pronto=>pronto,
        db_estado=> estad4
    );

    HEX1: hexa7seg port map(
        hexa =>conta4,
        sseg =>db_contagem
    );
```

```
        HEX2: hexa7seg port map(
            hexa =>memo4,
            sseg =>db_memoria
        );

        HEX3: hexa7seg port map(
            hexa =>estad4,
            sseg =>db_estado
        );
end architecture;
```

2.4) Plano de testes

#	Operação	Sinais de controle	Resultado
c.i.	Condições iniciais	clock=0 reset=0 iniciar=0 chaves=0000	pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=0000
1	“Resetar” circuito e observar a saída da memória	reset=1 iniciar=0 chaves=0000 clock ↑	pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=0000
2	Acionar sinal de clock 5 vezes com iniciar=0	reset=0 iniciar=0 chaves=0000 clock ↑ (5x)	(permanece no estado inicial A) pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=0000
3	Ajustar chaves para 0100, ativar iniciar=1 e acionar clock 1x	reset=0 iniciar=1 chaves=0100 clock ↑	(muda para estado B) pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=1011
4	Mantém chaves em 0100 e acionar clock 1x	reset=0 iniciar=0 chaves=0100 clock ↑	(muda para estado C) pronto=0 db_igual=0 db_contagem=0000

			db_memoria=0001 db_estado=1100
5	Mantém chaves em 0100 e acionar clock 1x	reset=0 iniciar=0 chaves=0100 clock ↑	(muda para estado D) pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=1101
6	Mantém chaves em 0100 e acionar clock 2x	reset=0 iniciar=0 chaves=0100 clock ↑ (2x)	(passa pelos estados C e D) pronto=0 db_igual=0 db_contagem=0001 db_memoria=0010 db_estado=1101
7	Mantém chaves em 0100 e acionar clock 2x	reset=0 iniciar=0 chaves=0100 clock ↑ (2x)	(passa pelos estados C e D) pronto=0 db_igual=1 db_contagem=0010 db_memoria=0100 db_estado=1101
8	Mantém chaves em 0100 e acionar clock 6x	reset=0 iniciar=0 chaves=0100 clock ↑ (6x)	(passa pelos estados C e D) pronto=0 db_igual=0 db_contagem=0101 db_memoria=1000 db_estado=1101
9	Ajustar chaves para 0001 e acionar clock 4x	reset=0 iniciar=0 chaves=0001 clock ↑ (4x)	(passa pelos estados C e D) pronto=0 db_igual=0 db_contagem=0111 db_memoria=0010 db_estado=1101
10	Ajustar chaves para 0010 e acionar clock 4x	reset=0 iniciar=0 chaves=0010 clock ↑ (4x)	(passa pelos estados C e D) pronto=0 db_igual=0 db_contagem=1001 db_memoria=0011 db_estado=1101
11	Ajustar chaves para 0100 e acionar clock 4x	reset=0 iniciar=0 chaves=0100 clock ↑ (4x)	(passa pelos estados C e D) pronto=0 db_igual=0 db_contagem=1011 db_memoria=1111 db_estado=1101
12	Ajustar chaves para 1000 e acionar clock 4x	reset=0 iniciar=0 chaves=1000 clock ↑ (4x)	(passa pelos estados C e D) pronto=0 db_igual=1 db_contagem=1101 db_memoria=1100 db_estado=1101
13	Ajustar chaves para 0000 e acionar clock 2x	reset=0	(muda para o estado E)

		iniciar=0 chaves=0000 clock ↑ (2x)	pronto=1 db_igual=1 db_contagem=1111 db_memoria=0000 db_estado=1110
14	Mantém chaves em 0000 e acionar clock 2x	reset=0 iniciar=0 chaves=0000 clock ↑ (2x)	(passa pelo estado E) pronto=0 db_igual=1 db_contagem=1111 db_memoria=0000 db_estado=0000
15	Mantém chaves em 0000 e acionar clock	reset=0 iniciar=0 chaves=0000 clock ↑	(termina no estado A) pronto=0 db_igual=1 db_contagem=1111 db_memoria=0000 db_estado=0000

2.5) Simulação do circuito

2.6) Designação de pinos

Sinal	Pino na Placa DE0-CV	Pino do FPGA	Analog Discovery
CLOCK	GPIO_0_D13	PIN_T22	StaticIO – Button 0/1 – DIO0
RESET	GPIO_0_D15	PIN_N19	StaticIO – Switch Push/Pull – DIO1
INICIAR	GPIO_0_D17	PIN_P19	StaticIO – Switch Push/Pull – DIO2
CHAVES(0)	GPIO_0_D19	PIN_P17	StaticIO – Switch Push/Pull – DIO3
CHAVES(1)	GPIO_0_D21	PIN_M18	StaticIO – Switch Push/Pull – DIO4
CHAVES(2)	GPIO_0_D23	PIN_L17	StaticIO – Switch Push/Pull – DIO5
CHAVES(3)	GPIO_0_D25	PIN_K17	StaticIO – Switch Push/Pull – DIO6
PRONTO	Led LEDR9	PIN_L1	-
DB_IGUAL	Led LEDR0	PIN_AA2	-
DB_CONTAGEM	Display HEX0	[0] PIN_U21 [1] PIN_V21 [2] PIN_W22 [3] PIN_W21 [4] PIN_Y22 [5] PIN_Y21 [6] PIN_AA22	-
DB_MEMORIA	Display HEX1	[0] PIN_AA20 [1] PIN_AB20 [2] PIN_AA19	-

		[3] PIN_AA18 [4] PIN_AB18 [5] PIN_AA17 [6] PIN_U22	
DB_ESTADO	Display HEX2	[0] PIN_Y19 [1] PIN_AB17 [2] PIN_AA10 [3] PIN_Y14 [4] PIN_V14 [5] PIN_AB22 [6] PIN_AB21	-

3) Elaboração do Circuito “circuito_exp4_ativ3”

3.1) Descrição VHDL do circuito “circuito_exp4_ativ3”

```
------------------------------------------------------------------
-- Arquivo   : unidade_controle.vhd
-- Projeto   : Experiencia 04 - Projeto de uma Unidade de Controle
------------------------------------------------------------------
-- Descricao : codificação da máquina de estados em VHDL
--             > para mais informação, consulte o livro
--               VHDL Descricao e Sintese de Circuitos Digitais
--               Roberto D'Amore, LTC Editora.
--               ou acesse a postagem do prof, Bruno Albertini em
--               https://balbertini.github.io/vhdl_fsm-pt_BR.html
------------------------------------------------------------------
-- Revisoes  :
--     Data        Versao  Autor             Descricao
--     21/01/2020  1.0     Edson Midorikawa  criacao
------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity unidade_controle is
  port (
    clock:     in  std_logic;
    reset:     in  std_logic;
    iniciar:   in  std_logic;
    fim:       in  std_logic;
    igual:     in std_logic;
```

```
    pronto:     out std_logic;
    zera:       out std_logic;
    conta:      out std_logic;
    acertou:    out std_logic;
    errou:      out std_logic;
    db_estado: out std_logic_vector(3 downto 0)
  );
end entity;

architecture fsm of unidade_controle is
  type t_estado is (A, B, C, D, E, F);
  signal Eatual, Eprox: t_estado;
begin
  -- memoria de estado
  process (clock,reset)
  begin
    if reset='1' then
      Eatual <= A;
    elsif clock'event and clock = '1' then
      Eatual <= Eprox;
    end if;
  end process;

  -- logica de proximo estado
  Eprox <=
      A when  Eatual=A and iniciar='0' else
      B when  Eatual=A and iniciar='1' else
      C when  Eatual=B else
      D when  Eatual=C and fim='0' and igual='1' else
      E when  Eatual=C and fim='1' else
      F when  Eatual=C and igual='0' else
      C when  Eatual=D else
      A when  Eatual=E else
      A when  Eatual=F else
      A;

  -- logica de saída (maquina de Moore)
  with Eatual select
    zera <=   '0' when A | C | D | E | F,
              '1' when B,
```

```
              '0' when others;

  with Eatual select
    conta <=  '0' when A | B | C | E | F,
              '1' when D,
              '0' when others;

  with Eatual select
    pronto <= '0' when A | B | C | D,
              '1' when E | F,
              '0' when others;

  with Eatual select
    acertou <='0' when A | B | C | D | F,
              '1' when E,
              '0' when others;

  with Eatual select
    errou <=  '0' when A | B | C | D | E,
              '1' when F,
              '0' when others;

  -- saida de depuracao (db_estado)
  with Eatual select
    db_estado <= "0000" when A,      -- 0
                 "1011" when B,      -- B
                 "1100" when C,      -- C
                 "1101" when D,      -- D
                 "1110" when E,      -- E
                 "1111" when F,      -- F
                 "0001" when others;  -- 1
end fsm;

-------------------------------------------------------------------
-- Arquivo : contador_163.vhd
-- Projeto : Experiencia 01 - Primeiro Contato com VHDL
-------------------------------------------------------------------
-- Descricao : contador binario hexadecimal (modulo 16)
-- similar ao CI 74163
-------------------------------------------------------------------
```

```
-- Revisoes :
-- Data Versao Autor Descricao
-- 29/12/2020 1.0 Edson Midorikawa criacao
----------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity contador_163 is -- entidade principal
 port (
clock : in std_logic; -- sinais de entrada
clr : in std_logic;
ld : in std_logic;
ent : in std_logic;
enp : in std_logic;
D : in std_logic_vector (3 downto 0);
Q : out std_logic_vector (3 downto 0); -- sinais de saída
rco : out std_logic
 );
end contador_163;
architecture comportamental of contador_163 is -- declaração da
arquitetura
 signal IQ: integer range 0 to 15;
begin
 process (clock,ent,IQ) -- inicio do process do circuito
 begin
if clock'event and clock='1' then
-- as mudanças no circuito ocorrem com o clock em 1
if clr='0' then IQ <= 0;
-- caso o sinal clear seja 0, a contagem é reiniciada
elsif ld='0' then IQ <= to_integer(unsigned(D));
-- caso o sinal load seja 0, a entrada D é carregada
elsif ent='1' and enp='1' then
-- ambos os sinais de controle precisam estar em 1
-- para que a contagem seja realizada
if IQ=15 then IQ <= 0;
-- caso chegue no final da contagem, volta p/ 0
else IQ <= IQ + 1;
-- caso contrário, soma-se 1 no contador
end if;
else IQ <= IQ;
```

```
-- caso um dos dois sinais de controle não esteja em nível
-- lógico alto, o contador permanece em seu estado atual
end if;
end if;
if IQ=15 and ent='1' then rco <= '1';
-- caso o contador tenha chegado no final, rco assume valor 1
else rco <= '0';
end if;
Q <= std_logic_vector(to_unsigned(IQ, Q'length));
-- a saída Q recebe o valor do sinal utilizado para a contagem
 end process; -- fim do process
end comportamental; -- fim da arquitetura




------------------------------------------------------------------
-- Arquivo   : comparador_85.vhd
-- Projeto   : Experiencia 02 - Um Fluxo de Dados Simples
------------------------------------------------------------------
-- Descricao : comparador binario de 4 bits
--           similar ao CI 7485
--           baseado em descricao criada por Edson Gomi (11/2017)
------------------------------------------------------------------
-- Revisoes  :
--  Data        Versao  Autor           Descricao
--  02/01/2021  1.0     Edson Midorikawa  criacao
------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity comparador_85 is  -- declaracao da entidade do comparador
  port (  -- entradas
    i_A3   : in  std_logic;
    i_B3   : in  std_logic;
    i_A2   : in  std_logic;
    i_B2   : in  std_logic;
    i_A1   : in  std_logic;
    i_B1   : in  std_logic;
```

```
    i_A0   : in  std_logic;
    i_B0   : in  std_logic;
    i_AGTB : in  std_logic;
    i_ALTB : in  std_logic;
    i_AEQB : in  std_logic;
    -- saidas
    o_AGTB : out std_logic;
    o_ALTB : out std_logic;
    o_AEQB : out std_logic
  );
end entity comparador_85;

architecture dataflow of comparador_85 is -- inicio da arquitetura
do comparador
  -- sinais intermediarios que comparam A e B sem levar em
consideracao as entradas de cascateamento
 signal agtb : std_logic;
  signal aeqb : std_logic;
  signal altb : std_logic;
begin
  -- equacoes dos sinais: pagina 462, capitulo 6 do livro-texto
  -- Wakerly, J.F. Digital Design - Principles and Practice, 4th
Edition
  -- veja tambem datasheet do CI SN7485 (Function Table)
  agtb <= (i_A3 and not(i_B3)) or
        (not(i_A3 xor i_B3) and i_A2 and not(i_B2)) or
        (not(i_A3 xor i_B3) and not(i_A2 xor i_B2) and i_A1 and
not(i_B1)) or
        (not(i_A3 xor i_B3) and not(i_A2 xor i_B2) and not(i_A1 xor
i_B1) and i_A0 and not(i_B0));
  -- checa se a > b
  aeqb <= not((i_A3 xor i_B3) or (i_A2 xor i_B2) or (i_A1 xor i_B1)
or (i_A0 xor i_B0));
  -- checa se a = b
  altb <= not(agtb or aeqb);
  -- checa se a < b
  o_AGTB <= agtb or (aeqb and (not(i_AEQB) and not(i_ALTB)));
  o_ALTB <= altb or (aeqb and (not(i_AEQB) and not(i_AGTB)));
  o_AEQB <= aeqb and i_AEQB;
```

```
  -- nas saidas, são levadas em consideracao as entradas de
cascateamento para obter um resultado final

end architecture dataflow; -- fim da arquitetura

-------------------------------------------------------------------
-- Arquivo   : ram_16x4.vhd
-- Projeto   : Experiencia 03 - Desenvolvendo o Fluxo de Dados
-------------------------------------------------------------------
-- Descricao : módulo de memória RAM 16x4
--             sinais we e ce ativos em baixo
--             codigo ADAPTADO do código encontrado no livro
--               VHDL Descricao e Sintese de Circuitos Digitais
--               Roberto D'Amore, LTC Editora.
-------------------------------------------------------------------
-- Revisoes  :
--     Data        Versao  Autor             Descricao
--     08/01/2020  1.0     Edson Midorikawa  criacao
-------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ram_16x4 is
   port (
       endereco     : in  std_logic_vector(3 downto 0);
       dado_entrada : in  std_logic_vector(3 downto 0);
       we           : in  std_logic;
       ce           : in  std_logic;
       dado_saida   : out std_logic_vector(3 downto 0)
    );
end entity ram_16x4;

architecture ram1 of ram_16x4 is
  type   arranjo_memoria is array(0 to 15) of std_logic_vector(3
downto 0);
  signal memoria : arranjo_memoria;
  attribute ram_init_file: string;
```

```
  attribute ram_init_file of memoria: signal is
"ram_conteudo_jogadas.mif";
begin

  process(ce, we, endereco, memoria)
  begin
    if ce = '0' then -- dado armazenado na subida de "we" com "ce=0"
      if rising_edge(we)
          then memoria(to_integer(unsigned(endereco))) <=
dado_entrada;
      end if;
    end if;
    dado_saida <= memoria(to_integer(unsigned(endereco)));
  end process;

end architecture ram1;









library ieee;
use ieee.std_logic_1164.all;

entity fluxo_dados is
    port (
        clock : in std_logic;

        zeraC : in std_logic;

        contaC : in std_logic;
        escreveM: in std_logic;
        chaves : in std_logic_vector (3 downto 0);
        fimC: out std_logic;
        db_contagem : out std_logic_vector (3 downto 0);
        db_memoria: out std_logic_vector(3 downto 0);
```

```
        igual: out std_logic
    );
end entity fluxo_dados;

architecture estrutural of fluxo_dados is

    component contador_163 is
        port (
        clock : in  std_logic;
        clr   : in  std_logic;
        ld    : in  std_logic;
        ent   : in  std_logic;
        enp   : in  std_logic;
        D     : in  std_logic_vector (3 downto 0);
        Q     : out std_logic_vector (3 downto 0);
        rco   : out std_logic
   );
    end component;



    component comparador_85 is
        port (  -- entradas
    i_A3   : in  std_logic;
    i_B3   : in  std_logic;
    i_A2   : in  std_logic;
    i_B2   : in  std_logic;
    i_A1   : in  std_logic;
    i_B1   : in  std_logic;
    i_A0   : in  std_logic;
    i_B0   : in  std_logic;
    i_AGTB : in  std_logic;
    i_ALTB : in  std_logic;
    i_AEQB : in  std_logic;
    -- saidas
    o_AGTB : out std_logic;
    o_ALTB : out std_logic;
    o_AEQB : out std_logic
        );
```

```
    end component;




    component ram_16x4 is
        port(
          endereco: in std_logic_vector(3 downto 0);
          dado_entrada: in std_logic_vector(3 downto 0);
          we: in std_logic;
          ce: in std_logic;
          dado_saida: out std_logic_vector(3 downto 0)
        );
    end component;




    signal
great,ZeraC_baixo,igual_out,menor,great_o,menor_o,enable_cin,rco_out
c: std_logic;
    signal contador_out, s_dado,s_endereco: std_logic_vector (3
downto 0);
    begin


          ZeraC_baixo<= not ZeraC;
        Contend: contador_163 port map (
            clock => clock,
            clr => ZeraC_baixo,
            ld    => '1',
            ent   => enable_Cin,
            enp   =>enable_Cin,
            D     => "0000",
            Q     => s_endereco,
            rco   => rco_outC
        );
      enable_Cin<=contaC;
      db_contagem<=s_endereco;
```

```
        compJog: comparador_85 port map (
            i_A3  =>s_dado(3),
              i_B3   => chaves(3),
              i_A2   =>s_dado(2),
              i_B2   => chaves(2),
              i_A1   =>s_dado(1),
              i_B1   => chaves(1),
              i_A0   =>s_dado(0),
              i_B0   => chaves(0),
              i_AGTB =>'0',
              i_ALTB => '0',
              i_AEQB => '1',
              -- saidas
              o_AGTB =>great,
              o_ALTB =>menor,
              o_AEQB => igual_out
        );


        memoria: ram_16x4 port map(
            endereco => s_endereco,
            dado_entrada => chaves,
            we => escreveM,
            ce => '0',
            dado_saida => s_dado
        );
        db_memoria<=s_dado;
     igual<=igual_out;
    fimC<=rco_outC;
    end architecture;


--------------------------------------------------------------------
-- Arquivo    : hexa7seg.vhd
-- Projeto    : Jogo do Desafio da Memoria
--------------------------------------------------------------------
-- Descricao : decodificador hexadecimal para
--             display de 7 segmentos
--
-- entrada: hexa - codigo binario de 4 bits hexadecimal
```

```
-- saida:   sseg - codigo de 7 bits para display de 7 segmentos
-------------------------------------------------------------------
-- dica de uso: mapeamento para displays da placa DE0-CV
--              bit 6 mais significativo é o bit a esquerda
--              p.ex. sseg(6) -> HEX0[6] ou HEX06
-------------------------------------------------------------------
-- Revisoes  :
--     Data        Versao  Autor             Descricao
--     29/12/2020  1.0     Edson Midorikawa  criacao
-------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

entity hexa7seg is
    port (
        hexa : in  std_logic_vector(3 downto 0);
        sseg : out std_logic_vector(6 downto 0)
    );
end hexa7seg;

architecture comportamental of hexa7seg is
begin

  sseg <= "1000000" when hexa="0000" else
          "1111001" when hexa="0001" else
          "0100100" when hexa="0010" else
          "0110000" when hexa="0011" else
          "0011001" when hexa="0100" else
          "0010010" when hexa="0101" else
          "0000010" when hexa="0110" else
          "1111000" when hexa="0111" else
          "0000000" when hexa="1000" else
          "0000000" when hexa="1000" else
          "0010000" when hexa="1001" else
          "0001000" when hexa="1010" else
          "0000011" when hexa="1011" else
          "1000110" when hexa="1100" else
          "0100001" when hexa="1101" else
          "0000110" when hexa="1110" else
```

```
          "0001110" when hexa="1111" else
          "1111111";

end comportamental;




library ieee;
use ieee.std_logic_1164.all;
entity circuito_exp4_ativ3 is
    port(
        clock : in std_logic;
        reset : in std_logic;
        iniciar : in std_logic;
        chaves : in std_logic_vector (3 downto 0);
        acertou : out std_logic;
        errou : out std_logic;
        pronto : out std_logic;
        db_igual : out std_logic;
        db_contagem : out std_logic_vector (6 downto 0);
        db_memoria : out std_logic_vector (6 downto 0);
        db_estado : out std_logic_vector (6 downto 0)
    );
end entity;

architecture estrutural of circuito_exp4_ativ3 is
    component fluxo_dados is
        port(
            clock : in std_logic;

            zeraC : in std_logic;

            contaC : in std_logic;
            escreveM: in std_logic;
            chaves : in std_logic_vector (3 downto 0);
            fimC: out std_logic;
            db_contagem : out std_logic_vector (3 downto 0);
            db_memoria: out std_logic_vector(3 downto 0);
```

```
        igual: out std_logic
    );
end component;

component unidade_controle is
    port(
        clock:     in  std_logic;
        reset:     in  std_logic;
        iniciar:   in  std_logic;
        fim:       in  std_logic;
        igual:     in std_logic;
        pronto:    out std_logic;
        zera:      out std_logic;
        conta:     out std_logic;
        acertou:   out std_logic;
        errou:     out std_logic;
        db_estado: out std_logic_vector(3 downto 0)
    );
end component;

component hexa7seg is
    port (
        hexa : in  std_logic_vector(3 downto 0);
        sseg : out std_logic_vector(6 downto 0)
    );
    end component;

signal conta4,memo4,estad4: std_logic_vector (3 downto 0);
signal conta,fim,zera,clk,igual_i: std_logic;
begin
    clk<=clock;
    FD: fluxo_dados port map(
        clock => clk,

        zeraC => zera,

        contaC => conta,
        escreveM=> '1',
```

```
            chaves => chaves,
            fimC=> fim,
            db_contagem => conta4,
            db_memoria=> memo4,
            igual=>igual_i
        );
        db_igual<=igual_i;
        UC: unidade_controle port map(
            clock=>clk,
            reset=>reset,
            iniciar=>iniciar,
            fim=>fim,
            zera=> zera,
            conta=> conta,
            pronto=>pronto,
            db_estado=> estad4,
            acertou =>acertou,
            igual=>igual_i,
            errou=>errou
        );

        HEX1: hexa7seg port map(
            hexa =>conta4,
            sseg =>db_contagem
        );

        HEX2: hexa7seg port map(
            hexa =>memo4,
            sseg =>db_memoria
        );

        HEX3: hexa7seg port map(
            hexa =>estad4,
            sseg =>db_estado
        );
end architecture;
```

3.2) Plano de testes

Teste 1 (Acertos)			
#	**Operação**	**Sinais de controle**	**Resultado**
c.i.	Condições iniciais	clock=0 reset=0 iniciar=0 chaves=0000	acertou=0 errou=0 pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=A
1	Resetar circuito	reset=1 iniciar=0 chaves=0000 clock ↑	acertou=0 errou=0 pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=A
2	Acionar sinal de clock 2 vezes com iniciar=0	reset=0 iniciar=0 chaves=0000 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=A
3	Setar chaves para 0001, ativar iniciar e acionar clock 1x	reset=0 iniciar=1 chaves=0001 clock ↑	acertou=0 errou=0 pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=B
4	Acionar clock 1x	reset=0 iniciar=0 chaves=0001 clock ↑	acertou=0 errou=0 pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=C
5	Acionar clock 1x	reset=0 iniciar=0 chaves=0001 clock ↑	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=0000 db_memoria=0001 db_estado=D
6	Setar chaves para 0010 e acionar clock 2x	reset=0 iniciar=0	acertou=0 errou=0

		chaves=0010 clock ↑ (2x)	pronto=0 db_igual=1 db_contagem=0001 db_memoria=0010 db_estado=D
7	Setar chaves para 0100 e acionar clock 2x	reset=0 iniciar=0 chaves=0100 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=0010 db_memoria=0100 db_estado=D
8	Setar chaves para 1000 e acionar clock 2x	reset=0 iniciar=0 chaves=1000 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=0011 db_memoria=1000 db_estado=D
9	Setar chaves para 0000 e acionar clock 2x	reset=0 iniciar=0 chaves=0000 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=0100 db_memoria=0000 db_estado=D
10	Setar chaves para 1000 e acionar clock 2x	reset=0 iniciar=0 chaves=1000 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=0101 db_memoria=1000 db_estado=D
11	Seta chaves para 0100 e acionar clock 2x	reset=0 iniciar=0 chaves=0100 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=0110 db_memoria=0100 db_estado=D
12	Setar chaves para 0010 e acionar clock 2x	reset=0 iniciar=0 chaves=0010 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=0111 db_memoria=0010 db_estado=D
13	Setar chaves para 0001 e acionar clock 2x	reset=0 iniciar=0 chaves=0001 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=1000 db_memoria=0001

			db_estado=D
14	Setar chaves para 0011 e acionar clock 2x	reset=0 iniciar=0 chaves=0011 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=1001 db_memoria=0011 db_estado=D
15	Setar chaves para 0111 e acionar clock 2x	reset=0 iniciar=0 chaves=0111 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=1010 db_memoria=0111 db_estado=D
16	Setar chaves para 1111 e acionar clock 2x	reset=0 iniciar=0 chaves=1111 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=1011 db_memoria=1111 db_estado=D
17	Setar chaves para 1110 e acionar clock 2x	reset=0 iniciar=0 chaves=1110 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=1100 db_memoria=1110 db_estado=D
18	Setar chaves para 1100 e acionar clock 2x	reset=0 iniciar=0 chaves=1100 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=1101 db_memoria=1100 db_estado=D
19	Setar chaves para 1000 e acionar clock 2x	reset=0 iniciar=0 chaves=1000 clock ↑ (2x)	acertou=0 errou=0 pronto=0 db_igual=1 db_contagem=1110 db_memoria=1000 db_estado=D
20	Setar chaves para 0000 e acionar clock 2x	reset=0 iniciar=0 chaves=0000 clock ↑ (2x)	acertou=1 errou=0 pronto=1 db_igual=1 db_contagem=1111 db_memoria=0000 db_estado=E

Teste 2 (Erro)			
#	**Operação**	**Sinais de controle**	**Resultado**
c.i.	Condições iniciais	clock=0 reset=0 iniciar=0 chaves=0000	acertou=0 errou=0 pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=0000
1	“Resetar” circuito e observar a saída da memória	reset=1 iniciar=0 chaves=0000 clock ↑	acertou=0 errou=0 pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=0000
2	Acionar sinal de clock 2 vezes com iniciar=0	reset=0 iniciar=0 chaves=0000 clock ↑ (2)	acertou=0 errou=0 pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=0000
3	Ajustar chaves para 0100, ativar iniciar=1 e acionar clock 2x	reset=0 iniciar=1 chaves=0001 clock ↑	acertou=0 errou=0 pronto=0 db_igual=0 db_contagem=0000 db_memoria=0001 db_estado=F

3.3) Simulação do circuito

3.4) Designação de pinos

Sinal	Pino na Placa DE0-CV	Pino do FPGA	Analog Discovery
CLOCK	GPIO_0_D13	PIN_T22	StaticIO – Button 0/1 – DIO0
RESET	GPIO_0_D15	PIN_N19	StaticIO – Switch Push/Pull – DIO1
INICIAR	GPIO_0_D17	PIN_P19	StaticIO – Switch Push/Pull – DIO2
CHAVES(0)	GPIO_0_D19	PIN_P17	StaticIO – Switch Push/Pull – DIO3
CHAVES(1)	GPIO_0_D21	PIN_M18	StaticIO – Switch Push/Pull – DIO4

CHAVES(2)	GPIO_0_D23	PIN_L17	StaticIO – Switch Push/Pull – DIO5
CHAVES(3)	GPIO_0_D25	PIN_K17	StaticIO – Switch Push/Pull – DIO6
PRONTO	Led LEDR9	PIN_L1	-
ACERTOU	Led LEDR8	PIN_L2	-
ERROU	Led LEDR7	PIN_U1	-
DB_IGUAL	Led LEDR0	PIN_AA2	-
DB_CONTAGEM	Display HEX0	[0] PIN_U21 [1] PIN_V21 [2] PIN_W22 [3] PIN_W21 [4] PIN_Y22 [5] PIN_Y21 [6] PIN_AA22	-
DB_MEMORIA	Display HEX1	[0] PIN_AA20 [1] PIN_AB20 [2] PIN_AA19 [3] PIN_AA18 [4] PIN_AB18 [5] PIN_AA17 [6] PIN_U22	-
DB_ESTADO	Display HEX2	[0] PIN_Y19 [1] PIN_AB17 [2] PIN_AA10 [3] PIN_Y14 [4] PIN_V14 [5] PIN_AB22 [6] PIN_AB21	-