



**Laboratório Digital I - PCS3635**

**Planejamento da Experiência 5:**

**Considerações de Projeto de Circuitos em FPGA**

Marco Aurélio C. O. Prado - NUSP 11257605

Victor Hoefling Padula - NUSP 10770051

Turma 04 - Bancada A1

São Paulo - SP

09/02/2021

## 1) Objetivo

Os objetivos da aula consistem em aprender sobre:

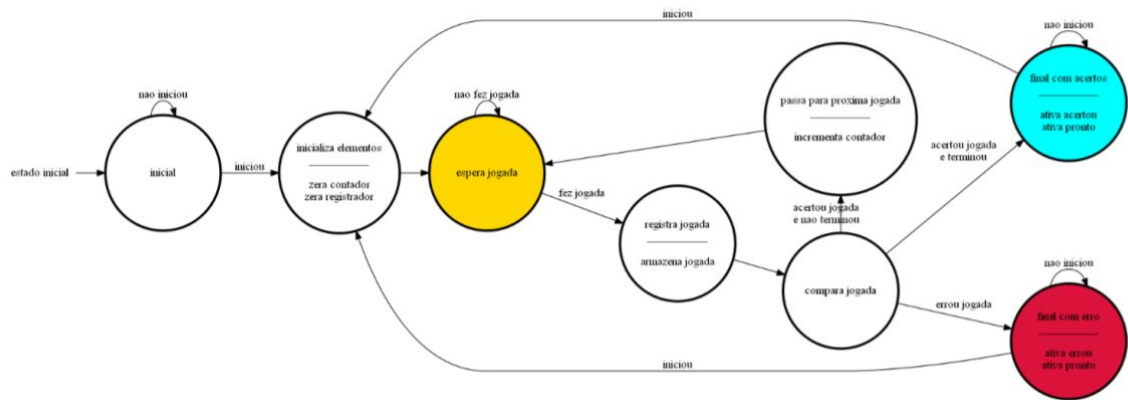
- Projeto de circuitos usando descrição estrutural VHDL;
- Interface de circuitos digitais com elementos externos de entrada de dados;
- Documentação de projetos (planejamento e relatório);
- Uso de sinais periódicos como clock;

## 2) Elaboração do Circuito “circuito\_exp5.vhd”

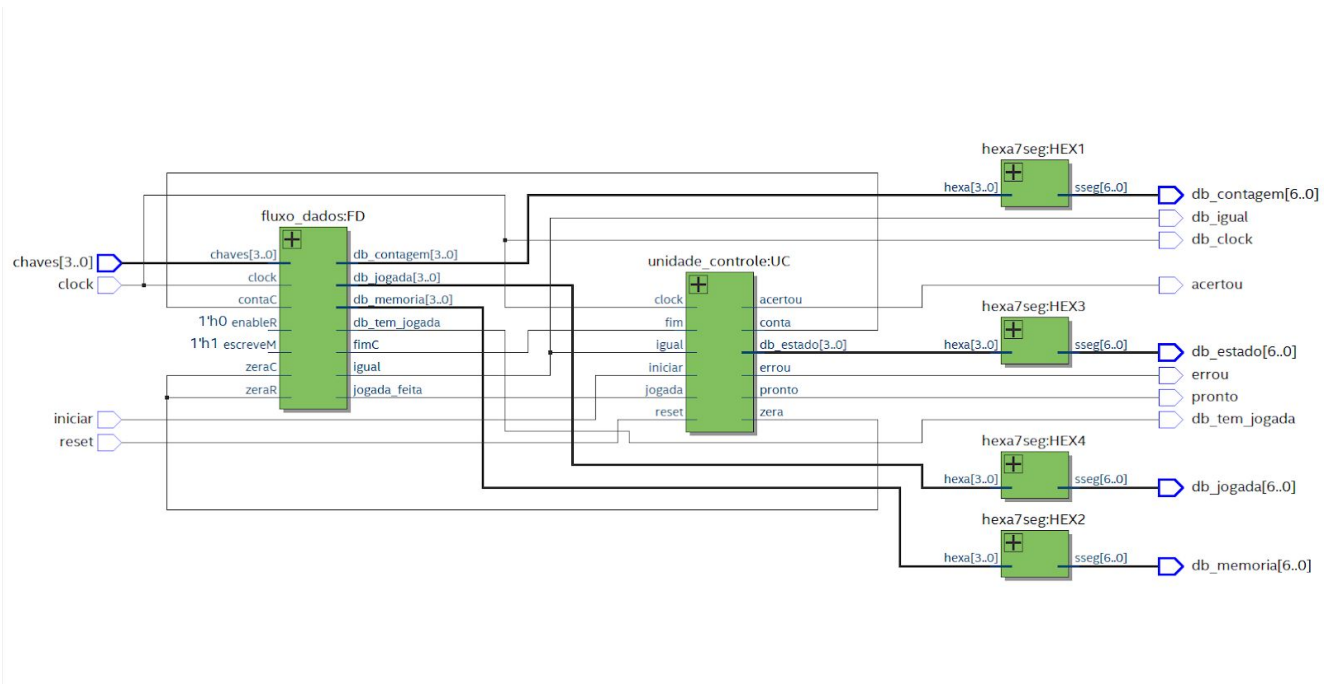
### 2.1) Descrição do funcionamento do circuito

O circuito é composto de uma unidade de controle e um fluxo de dados com 1 contador, 1 comparador, 1 memória, 1 registrador, conversores para 7 segmentos e 1 detector de jogadas. Ele segue a lógica de controle de começar em um estado “A” de espera pelo sinal de início. A partir daí, segue para o estado “B” que zera o contador e o registrador, e parte para o estado “J” de espera de uma jogada, pela mudança nos sinais “chaves”. Depois, segue para um estado “M” de atualização da memória, endereçada pela saída do contador. Em seguida, vai para um estado “C” de comparação, em que o comparador avalia se a entrada das chaves é igual ao conteúdo da memória no endereço indicado pelo contador. Se for igual, o circuito continua, senão vai para o estado de erro “F” e permanece nele até um sinal de início, podendo voltar ao estado “B”. Se o circuito continua, ele segue para o estado “D”, onde incrementa o contador e conseqüentemente o endereço da memória, e volta para o estado “J”. Se as jogadas continuarem iguais ao conteúdo da memória até o fim dos endereços, o circuito vai para o estado de acerto “E”, e permanece nele até um sinal de início, podendo voltar ao estado “B”. As saídas de acerto e erro se acendem nos estados “E” e “F”, e as saídas de depuração mostram o conteúdo da memória, a saída do contador, o estado atual e a jogada atual em displays de 7 segmentos. A saída “tem jogada” acende se houver mudança nas chaves, e a entrada reset leva o circuito ao estado “A”. Além disso, a saída de depuração do clock copia o sinal de clock colocado na entrada clock. Para simplificação, os seguintes diagramas podem ser usados para entender a lógica do funcionamento do circuito:

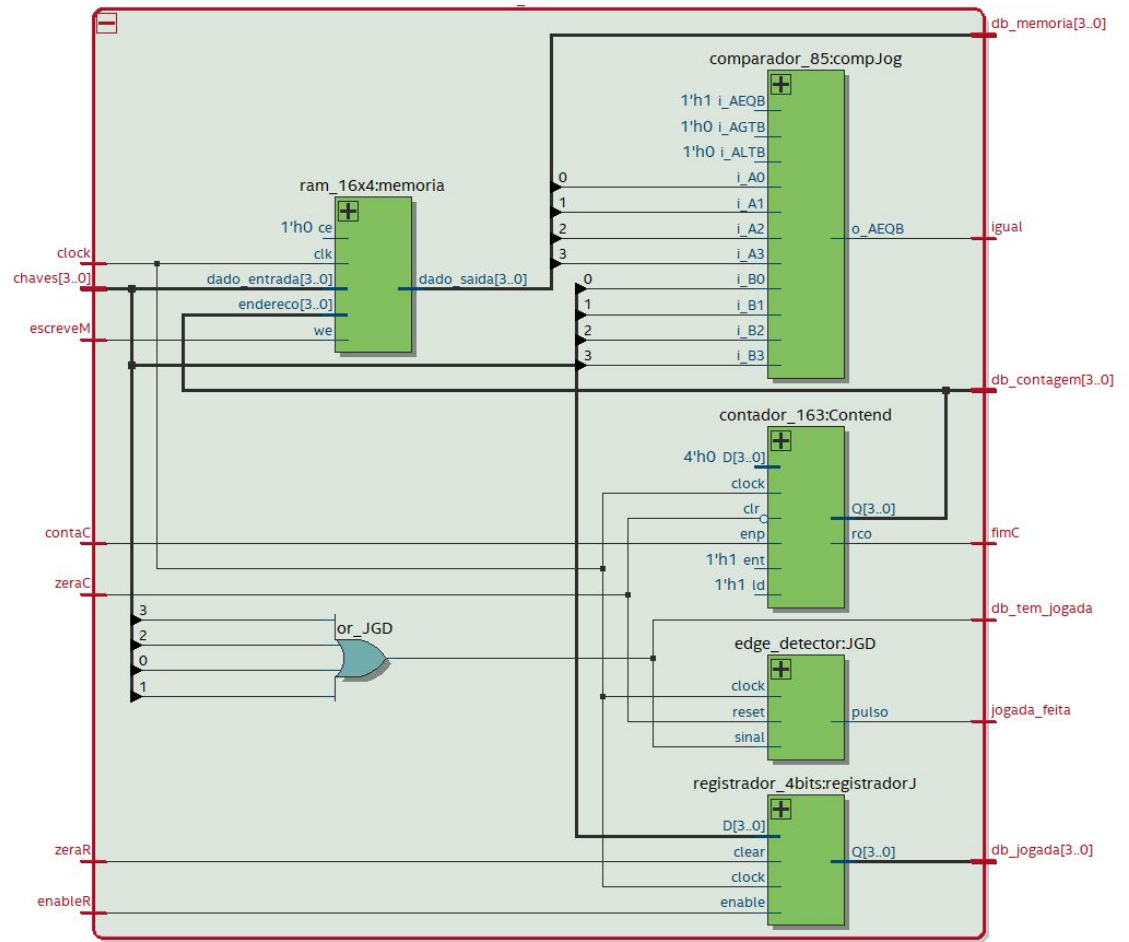
Máquina de estados finita:



Visão a nível RTL do circuito:



Visão a nível RTL do Fluxo de Dados:



## 2.2) Descrição VHDL do circuito “circuito\_exp5.vhd”

```
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;
    jogada:     in  std_logic;
    zera:       out std_logic;
    conta:      out std_logic;
    pronto:     out std_logic;
    db_estado:  out std_logic_vector(3 downto 0);
```

```

    acertou: out std_logic;
    errou: out std_logic;
    registra: out std_logic;
    igual: in std_logic
);
end entity;

architecture fsm of unidade_controle is
    type t_estado is (A, B, C, D, E, F, M, J);
    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
        J when Eatual=B else
        J when Eatual=J and jogada='0' else
        M when Eatual=J and jogada='1' else
        C when Eatual=M else
        D when Eatual=C and fim='0' and igual='1' else
        J when Eatual=D else
        E when Eatual=C and fim='1' else
        F when Eatual=C and igual='0' else
        M when Eatual=D else
        A when Eatual=E else
        B when Eatual=E and iniciar='1' else
        E when Eatual=E and iniciar='0' else
        B when Eatual=F and iniciar='1' else
        F when Eatual=F and iniciar='0' else
        A;

```

```

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

with Eatual select

    zero <=     '0' when A | C | D | E | F | M | J,
                '1' when B,
                '0' when others;

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

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

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

with Eatual select
    errou <=    '0' when A | B | C | D | E | M | J,
                '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
                "0111" when M, -- M

```

```

        "1000" when J, -- J
        "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
-----
-- Descriçao   : comparador binario de 4 bits
--               similar ao CI 7485
--               baseado em descricao criada por Edson Gomi (11/2017)
-----
-- Revisoes    :
-- Data        Versao  Autor            Descriçao
-- 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 cascadeamento
    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 cascadeamento
para obter um resultado final

```

```

end architecture dataflow; -- fim da arquitetura

```

```

-----
-- Arquivo      : ram_16x4.vhd
-- Projeto      : Experiencia 05 - Consideracoes de Projeto com VHDL
-----
-- Descricao    : módulo de memória RAM sincrona 16x4
--                sinais we e ce ativos em baixo
--                codigo ADAPTADO do código encontrado no livro
--                VHDL Descricao e Sintese de Circuitos Digitais
--                de Roberto D'Amore, LTC Editora.
-----
-- Revisoes     :
--
--      Data      Versao  Autor          Descricao
--      08/01/2020  1.0    Edson Midorikawa  criacao
--      01/02/2020  2.0    Antonio V.S.Neto  Atualizacao para
--                                     RAM sincrona para
--                                     minimizar problemas
--                                     com Quartus.
--      02/02/2020  2.1    Edson Midorikawa  revisao de codigo e
--                                     arquitetura para
--                                     simulacao com ModelSim
-----

```

```

library ieee;
use ieee.std_logic_1164.all;

```

```

use ieee.numeric_std.all;

entity ram_16x4 is
    port (
        clk          : in  std_logic;
        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 ram_mif of ram_16x4 is
    type arranjo_memoria is array(0 to 15) of std_logic_vector(3 downto 0);
    signal memoria : arranjo_memoria;

    -- Configuracao do Arquivo MIF
    attribute ram_init_file: string;
    attribute ram_init_file of memoria: signal is
"ram_conteudo_jogadas.mif";

begin

    process(clk)
    begin
        if (clk = '1' and clk'event) then
            if ce = '0' then -- dado armazenado na subida de "we" com "ce=0"

                -- Detecta ativacao de we (ativo baixo)
                if (we = '0')
                    then memoria(to_integer(unsigned(endereco))) <=
dado_entrada;
                    end if;

                end if;
            end if;
        end process;
    end process;
end process;

```

```

-- saida da memoria
dado_saida <= memoria(to_integer(unsigned(endereco)));

end architecture ram_mif;

-- Dados iniciais (para simulacao com Modelsim)
architecture ram_modelsim of ram_16x4 is
    type    arranjo_memoria is array(0 to 15) of std_logic_vector(3 downto
0);
    signal memoria : arranjo_memoria := (
                                "0001",
                                "0010",
                                "0100",
                                "1000",
                                "0100",
                                "0010",
                                "0001",
                                "0001",
                                "0010",
                                "0010",
                                "0100",
                                "0100",
                                "1000",
                                "1000",
                                "0001",
                                "0100" );

begin

    process(clk)
    begin
        if (clk = '1' and clk'event) then
            if ce = '0' then -- dado armazenado na subida de "we" com "ce=0"

                -- Detecta ativacao de we (ativo baixo)
                if (we = '0')
                    then memoria(to_integer(unsigned(endereco))) <=
dado_entrada;
                end if;
            end if;
        end if;
    end process;
end architecture ram_modelsim;

```

```

        end if;
    end if;
end process;

-- saida da memoria
dado_saida <= memoria(to_integer(unsigned(endereco)));

end architecture ram_modelsim;

-----

-- Arquivo      : edge_detector.vhd
-- Projeto      : Experiencia 05 - Consideracoes de Projeto com FPGA
-----

-- Descricao : detector de borda
--             gera um pulso na saida de 1 periodo de clock
--             a partir da detecao da borda de subida sa entrada
--             sinal de reset ativo em alto
--             > adaptado a partir de codigo VHDL disponivel em
--             https://surf-vhdl.com/how-to-design-a-good-edge-detector/
-----

-- Revisoes  :
--   Data      Versao  Autor          Descricao
--   29/01/2020  1.0    Edson Midorikawa  criacao
-----

library ieee;
use ieee.std_logic_1164.all;

entity edge_detector is
port (
    clock : in  std_logic;
    reset  : in  std_logic;
    sinal  : in  std_logic;
    pulso  : out std_logic);
end edge_detector;

architecture rtl of edge_detector is
    signal reg0 : std_logic;
    signal reg1 : std_logic;

```

```

begin

    detector : process(clock,reset)
    begin
        if(reset='1') then
            reg0 <= '0';
            reg1 <= '0';

            elsif(rising_edge(clock)) then
                reg0 <= sinal;
                reg1 <= reg0;
            end if;
        end process;

        pulso <= not reg1 and reg0;

```

```

end rtl;

```

```

-----
-- Arquivo      : registrador_4bits.vhd
-- Projeto      : Experiencia 05 - Consideracoes de Projeto com FPGA
-----
-- Descricao    : registrador de 4 bits
--                com clear assincrono e enable
-----
-- Revisoes     :
--      Data      Versao  Autor           Descricao
--      31/01/2020 1.0    Edson Midorikawa  criacao
-----

```

```

library ieee;
use ieee.std_logic_1164.all;

entity registrador_4bits is
    port (
        clock: in  std_logic;
        clear: in  std_logic;
        enable: in  std_logic;
        D:      in  std_logic_vector(3 downto 0);
        Q:      out std_logic_vector(3 downto 0)
    );
end entity;

```

```

);
end entity;

architecture arch of registrador_4bits is
    signal IQ: std_logic_vector(3 downto 0);
begin
    process(clock, clear, IQ)
    begin
        if (clear = '1') then IQ <= (others => '0');
        elsif (clock'event and clock='1') then
            if (enable='1') then IQ <= D; end if;
        end if;
    end process;

    Q <= IQ;
end architecture;

library ieee;
use ieee.std_logic_1164.all;

entity fluxo_dados is
    port (
        clock : in std_logic;
        zeraC : in std_logic;
        zeraR: in std_logic;
        enableR: in std_logic;
        contaC : in std_logic;
        escreveM: in std_logic;
        chaves : in std_logic_vector (3 downto 0);
        fimC: out std_logic;
        db_tem_jogada: out std_logic;
        db_contagem : out std_logic_vector (3 downto 0);
        db_memoria: out std_logic_vector(3 downto 0);
        jogada_feita: out std_logic;
        db_jogada: 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(
        clk          : in  std_logic;
        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;

component registrador_4bits is
    port (
        clock: in  std_logic;
        clear: in  std_logic;
        enable: in  std_logic;
        D: in  std_logic_vector(3 downto 0);
        Q: out std_logic_vector(3 downto 0)
    );
end component;

component edge_detector is
    port (
        clock : in  std_logic;
        reset  : in  std_logic;
        sinal  : in  std_logic;
        pulso  : out std_logic);
end component;

signal
great,ZeraC_baixo,igual_out,menor,great_o,menor_o,enable_cin,rco_outc,or_J
GD: 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   => '1',
        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(
        clk=> clock,
        endereco => s_endereco,
        dado_entrada => chaves,
        we => escreveM,
        ce => '0',
        dado_saida => s_dado
    );
    db_memoria<=s_dado;
    igual<=igual_out;

```

```
fimC<=rco_outC;
```

```
registradorJ: registrador_4bits port map(  
  clock=>clock,  
  clear=>zeraR,  
  enable=>enableR,  
  D=>chaves,  
  Q=>db_jogada);
```

```
or_JGD<=chaves(0) or chaves(1) or chaves(2) or chaves(3);  
JGD: edge_detector port map(  
  clock=>clock,  
  reset=>zeraC,  
  sinal=>or_JGD,  
  pulso=>jogada_feita  
);  
db_tem_jogada<=or_JGD;
```

```
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_exp5 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);
        db_jogada : out std_logic_vector (6 downto 0);
        db_clock : out std_logic;
        db_tem_jogada : out std_logic

    );
end entity;

```

architecture estrutural of circuito\_exp5 is

```

    component fluxo_dados is
        port(
            clock : in std_logic;
            zeraC : in std_logic;
            zeraR: in std_logic;
            enableR: in std_logic;
            contaC : in std_logic;
            escreveM: in std_logic;
            chaves : in std_logic_vector (3 downto 0);
            fimC: out std_logic;
            db_tem_jogada: out std_logic;
            db_contagem : out std_logic_vector (3 downto 0);
            db_memoria: out std_logic_vector(3 downto 0);
            jogada_feita: out std_logic;
            db_jogada: 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;
        registra:  out std_logic;
        jogada:    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, joga4: std_logic_vector (3 downto 0);
signal conta, fim, zera, registra, clk, jogada, igual_i: std_logic;
begin
    clk<=clock;
    FD: fluxo_dados port map(
        clock => clk,
        zeraR=> zera,
        zeraC => zera,
        db_jogada=>joga4,
        enableR=> registra,
        contaC => conta,
        escreveM=> '1',
        chaves => chaves,
        fimC=> fim,
        db_tem_jogada=>db_tem_jogada,
        jogada_feita=>jogada,
        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,
        jogada=>jogada,
        db_estado=> estad4,
        acertou =>acertou,
        igual=>igual_i,
        errou=>errou
    );
    db_clock<=clk;
    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
    );

    HEX4: hexa7seg port map(
        hexa=>joga4,
        sseg=> db_jogada
    );
end architecture;
```

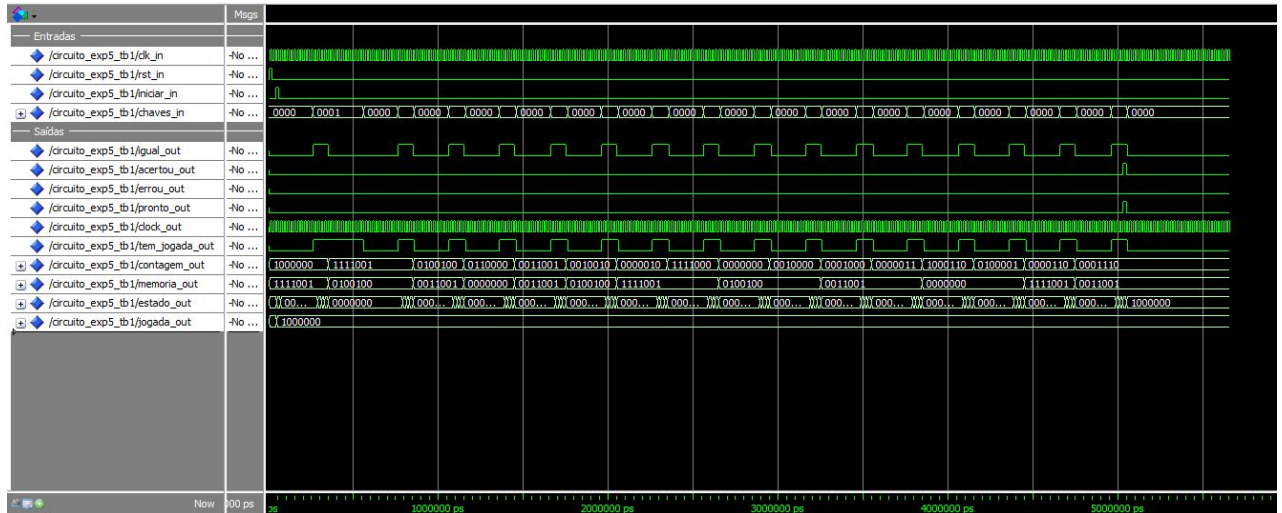
## 2.3) Plano de testes

Cenário #1 – Acerto das 16 jogadas				
#	Operação	Sinais de entrada	Resultado esperado	Resultado observado
c.i.	Condições Iniciais	--		
1	“Resetar” circuito	acionar reset		
2	Aguardar alguns segundos	--	Circuito zerado	Circuito zerado
3	Acionar sinal iniciar	acionar iniciar	Circuito muda p/ estado B	Circuito muda p/ estado B
4	acionar primeira entrada (jogada 1)	acionar chave(0)	Igual ativo Tem_jogada ativo	Igual ativo Tem_jogada ativo
5	acionar segunda entrada (jogada 2)	acionar chave(1)	Igual ativo Tem_jogada ativo	Igual ativo Tem_jogada ativo
...				
19	aciona última entrada (jogada 16)	aciona chave(2)	saídas pronto e acertou ativadas	
Cenário #2 – Acerto das 4 primeiras jogadas e erro na 5ª jogada				
#	Operação	Sinais de entrada	Resultado esperado	Resultado observado
c.i.	Condições Iniciais	--		
1	“Resetar” circuito	acionar reset		
2	Aguardar alguns segundos	--	Circuito zerado	Circuito zerado
3	Acionar sinal iniciar	acionar iniciar	Circuito muda p/ estado B	Circuito muda p/ estado B
4	acionar primeira entrada (jogada 1)	acionar chave(0)	Igual ativo Tem_jogada ativo	Igual ativo Tem_jogada ativo
5	acionar segunda entrada (jogada 2)	acionar chave(1)	Igual ativo Tem_jogada ativo	Igual ativo Tem_jogada ativo
...				
8	aciona quinta entrada (jogada 5)	aciona chave(0)	saídas pronto e errou ativadas	

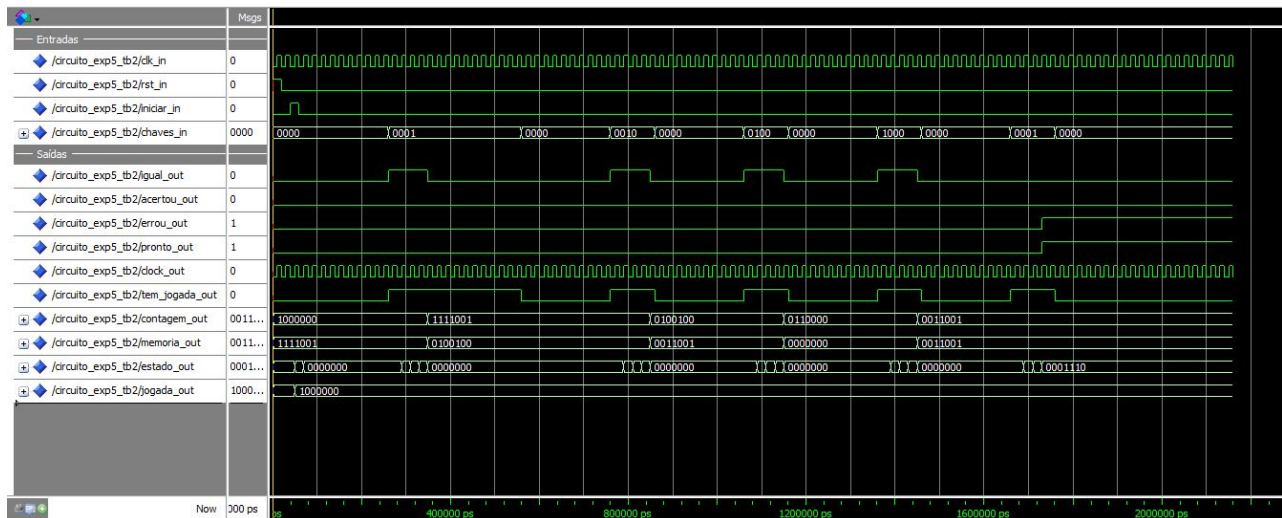


## 2.4) Simulação do circuito no ModelSim

### 2.4.1) Cenário #1



### 2.4.2) Cenário #2



## 2.5) Designação de pinos

Sinal	Pino na Placa DE0-CV	Pino no FPGA	Analog Discovery
CLOCK	GPIO_0_D13	PIN_T22	StaticIO – LED – DIO0 Patterns – Clock – 1kHz
RESET	GPIO_0_D15	PIN_N19	StaticIO – Button 0/1 – DIO1
INICIAR	GPIO_0_D17	PIN_P19	StaticIO – Button 0/1 – DIO2
CHAVES(0)	GPIO_0_D19	PIN_P17	StaticIO – Button 0/1 – DIO3
CHAVES(1)	GPIO_0_D21	PIN_M18	StaticIO – Button 0/1 – DIO4
CHAVES(2)	GPIO_0_D23	PIN_L17	StaticIO – Button 0/1 – DIO5
CHAVES(3)	GPIO_0_D25	PIN_K17	StaticIO – Button 0/1 – DIO6
PRONTO	Led LEDR9	PIN_L1	-
ACERTOU	Led LEDR8	PIN_L2	-
ERROU	Led LEDR7	PIN_U1	-
DB_IGUAL	Led LEDR0	PIN_AA2	-
DB_CLOCK	Led LEDR1	PIN_AA1	-
DB_TEM_JOGADA	Led LEDR2	PIN_W2	-
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_JOGADA	Display HEX2	[0] PIN_Y19 [1] PIN_AB17 [2] PIN_AA10 [3] PIN_Y14 [4] PIN_V14 [5] PIN_AB22 [6] PIN_AB21	-

DB_ESTADO	Display HEX5	[0] PIN_N9 [1] PIN_M8 [2] PIN_T14 [3] PIN_P14 [4] PIN_C1 [5] PIN_C2 [6] PIN_W19	-
-----------	--------------	---	---