Laboratório Digital I - PCS3635

Planejamento da Experiência 8:

Projeto de Extensão do Jogo do Desafio da Memória (Semana 1)

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

São Paulo - SP
16/03/2021

1) Objetivo

Os objetivos da aula consistem em aprender sobre:

- Projeto de circuitos usando descrição estrutural VHDL;
- Documentação de projetos (planejamento e relatório);
- Síntese de circuitos para uma placa FPGA

2) Elaboração do Circuito “circuito_semana1.vhd”

2.1) Descrição do funcionamento do circuito “circuito_semana1.vhd”

Este circuito consiste na versão completa do Jogo do Desafio da Memória, apresentando um ciclo completo de jogadas onde a cada rodada é acrescentado um valor a mais na memória, ou seja, a cada rodada o número de jogadas aumenta. Caso o jogador erre, ele pode acionar o botão “repete”, onde a sequência de jogadas correta é exibida para que ele saiba onde errou. Além disso, o circuito desenvolvido nesta semana conta com uma funcionalidade adicional: a de selecionar níveis de dificuldade diferentes para o jogo através do sinal de 2 bits “nivel”. O jogo possui 4 níveis de dificuldade: 00, 01, 10 e 11, onde o jogador precisa acertar, respectivamente, 4, 8, 12 e 16 jogadas para ganhar.

Para iniciar o jogo, deve-se apertar o botão “iniciar”. Após isso, o circuito aguardará pela primeira jogada, que deve ser inserida pelo jogador através dos botões ligados aos sinais “botoes”. Após isso, o jogador deve repetir a primeira jogada e acrescentar mais uma. Na rodada seguinte, o jogador deve repetir as 2 primeiras jogadas e acrescentar uma terceira. O ciclo continua até que o jogador erre alguma das jogadas ou demore mais do que 5 segundos para fazer uma jogada. Esse *timeout* começa a funcionar a partir da primeira rodada.

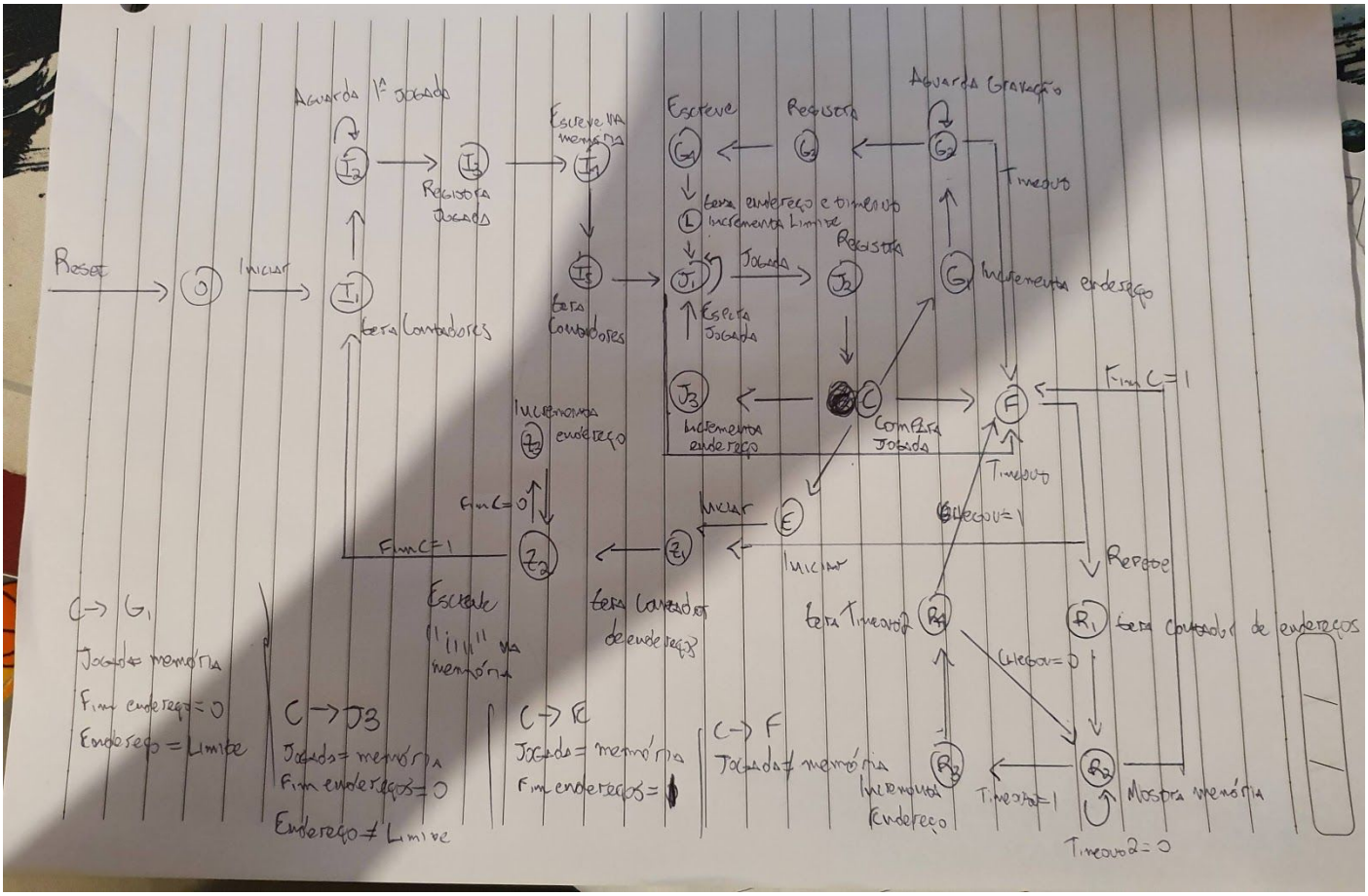
A lógica de estados é a mesma do experimento anterior, mas foi feita uma refatoração no nome dos estados. Para implementação dos níveis, foram incluídas as chaves “níveis” e foi feita uma pequena modificação no fluxo de dados. Foi incluído nele um comparador, que compara a saída do contador com um sinal chamado de “final”. O sinal final é determinado pela entrada “nível”, que através de um multiplexador alterna entre 4,8,12 e 16 se a entrada for 00, 01, 10 e 11 respectivamente. A saída de fim do contador de endereços do fluxo de dados agora recebe a saída “=” do comparador, ou seja, quando a saída do contador for igual ao “final” determinado pelo nível, o fluxo de dados envia para a unidade de controle um sinal dizendo que o contador de endereços chegou ao fim. Dessa forma, não foi necessária nenhuma mudança na máquina de estados.

O circuito segue a seguinte lógica de estados: primeiro ele passa pelo estado A ao ser acionada a entrada Reset, em seguida, vai para o estado I1, onde reinicia todos os contadores. Depois, vai para o estado I2, onde aciona o

led Espera e aguarda a primeira jogada a ser gravada. Então segue para os estados de lógica de gravação, I3, onde registra a jogada, I4, onde guarda a jogada na memória, e depois I5, onde zera o contador de endereços e do timeout.Depois, entra no loop do jogo, indo para o estado J1, onde aciona o contador timeout, que conta 5000 ciclos de clock (para um clock de 1 khz isso equivale a 5 segundos), e aciona o led Espera, e permanece nele até ser feita uma jogada ou acabar o tempo. Se for feita uma jogada antes de acabar o tempo, ele vai para o estado J2, se o tempo acabar ele vai para o estado "perdeu", ou F. No estado J2, ele registra a jogada feita e vai para o estado de comparação C. No estado C, ele pode ou ter errado a jogada (nesse caso ele vai para o estado F), ou acertado. Se ele acertou e o endereço não é igual ao limite, ele vai para o estado J3; se o endereço é igual ao limite mas o contador de endereços não está no fim, ele vai para o estado G1; se ele acertou e estiver no último endereço da memória ele vai para o estado E (ganhou). No estado J3 ele incrementa o contador de endereços, zera o timeout e volta para o estado J1. no estado G1 ele também incrementa o contador de endereços e zera o timeout, mas vai para o estado G2. No estado G2 ele aguarda a próxima jogada, aciona o timeout de forma similar ao J1, mas se o jogador fizer uma jogada a tempo ele vai para os estados de lógica de gravação, G3, onde registra a jogada, G4, onde guarda a jogada na memória, e depois L, onde incrementa o limite e zera o contador de endereços e do timeout. Se der timeout ele vai para o estado F (perdeu). Nos estados E e F o circuito permanece neles a menos que seja acionada a entrada Iniciar, onde ele volta para o estado I1. O desafio da aula passada consistiu em implementar a função "repete", que mostra todas as jogadas salvas em memória se o jogador perder e apertar o botão Repete. Para tanto, acrescentamos os estados R1,R2,R3 e R4, que acionam a lógica de repetição, e os estados Z1,Z2 e Z3, que acionam a lógica de reset da memória. Na lógica de repetição, quando o jogador perde e aciona o botão repete, ele vai para o estado R1, que zera o contador de endereços. Em seguida vai para o estado R2, onde aguarda 2 segundos antes de ir pro estado R3. No estado R3 ele incrementa o endereço e vai para o estado R4, onde zera o contador de tempo do estado R2. Se ele estiver no estado R2 e chegar ao fim da memória ele termina de mostrar e para no estado de erro F, onde pode-se repetir a exibição. E se estiver no estado R4 e tiver chegado a uma região da memória com "1111" salvo ele também para a exibição e vai para o estado F. A exibição é feita por meio dos LEDS da placa. A lógica de reset ocorre quando o jogador perde e aperta Iniciar novamente. Ao fazer isso o circuito vai para o estado Z1, onde zera o contador de endereços e vai para o estado Z2, onde aciona a saída reset_m que direciona "1111" para a entrada de escrita da memória, e aciona o sinal de escrita. Depois, se ele não chegou ao final da memória, ele segue para

o estado Z3, onde incrementa o endereço e volta para o estado Z2. Se ele estiver no Z2 e chegar ao fim da memória ele volta para o início da máquina de estados.

O diagrama a seguir ilustra o funcionamento da máquina de estados:



2.1) Descrição VHDL do circuito “circuito_semana1.vhd”

Seguem os códigos vhdl dos componentes que não foram alterados da semana passada: contador, contador modificado para timeout, contador para timeout de 2 segundos, comparador, memória, registrador, detector de borda e conversor de hexadecimal para 7 segmentos, respectivamente.

```
-----------------------------------------------------------------
-- 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 : contador_163_mod.vhd
-- Projeto : Experiencia 05
-----------------------------------------------------------------
-- Descricao : contador binario hexadecimal (modulo 16)
-- similar ao CI 74163
-----------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity contador_mod 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 (12 downto 0);
Q : out std_logic_vector (12 downto 0); -- sinais de saída
rco : out std_logic
 );
end contador_mod;
architecture comportamental of contador_mod is -- declaração da
arquitetura
 signal IQ: integer range 0 to 8191;
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>=5000 then IQ <= 5000;
-- 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>=5000 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 : contador_163_mod.vhd
-- Projeto : Experiencia 05
-------------------------------------------------------------------
-- Descricao : contador binario hexadecimal (modulo 16)
-- similar ao CI 74163
-------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
```

```
use IEEE.numeric_std.all;
entity contador_mod_2s 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 (10 downto 0);
Q : out std_logic_vector (10 downto 0); -- sinais de saída
rco : out std_logic
 );
end contador_mod_2s;
architecture comportamental of contador_mod_2s is -- declaração da
arquitetura
 signal IQ: integer range 0 to 8191;
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>=2000 then IQ <= 2000;
-- 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>=2000 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 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_inicial.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;

  -- 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 := (
                                        "1111",
                                        "1111",
                                        "1111",
                                        "1111",
                                        "1111",
                                        "1111",
                                        "1111",
                                        "1111",
                                        "1111",
```

```
                                        "1111",
                                        "1111",
                                        "1111",
                                        "1111",
                                        "1111",
                                        "1111",
                                        "1111" );

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;

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

end architecture ram_modelsim;
```

```
-------------------------------------------------------------------
-- 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;

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;
```

```
----------------------------------------------------------------------
------
-- 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;
```

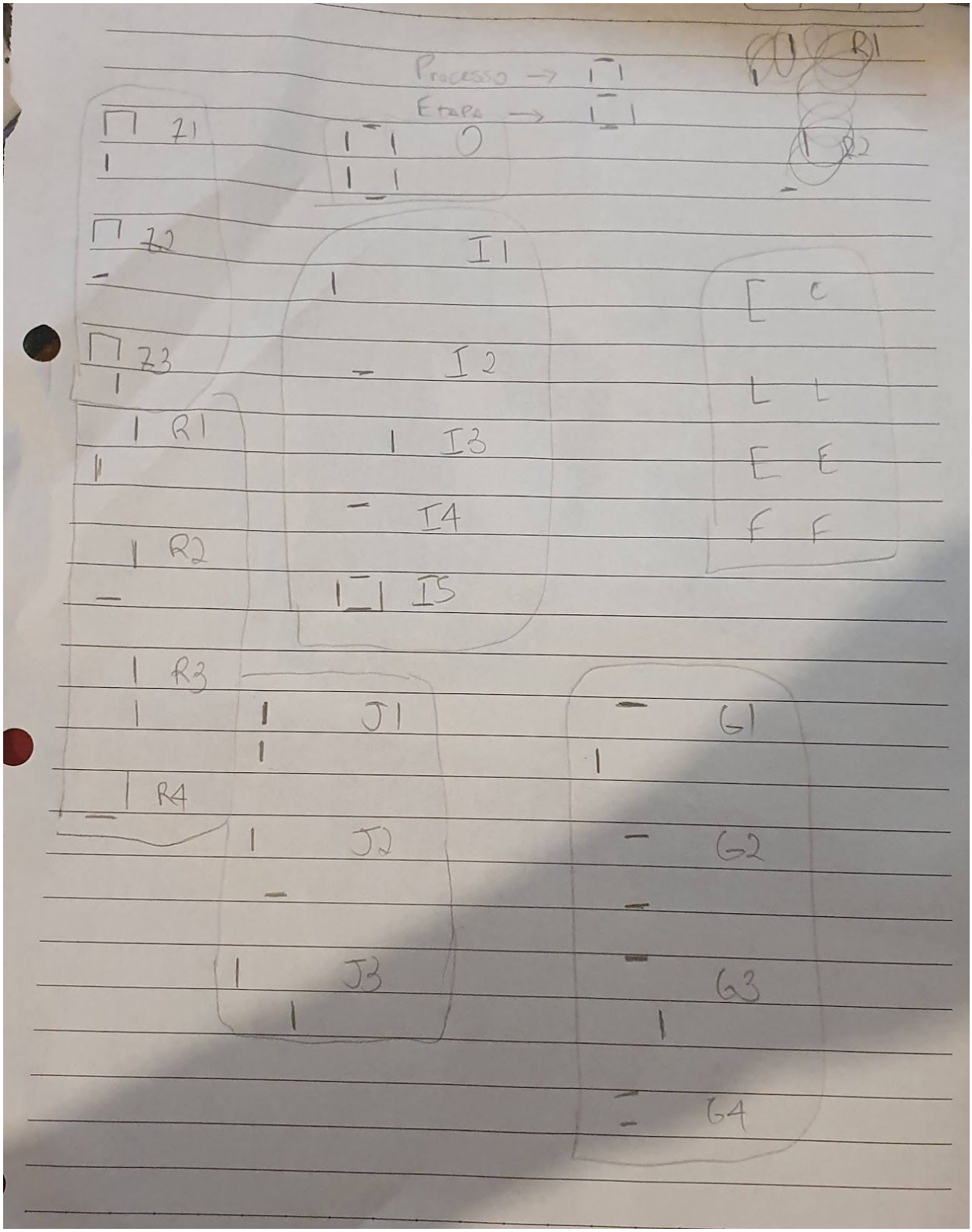
```
  end process;

  pulso <= not reg1 and reg0;

end rtl;
```

```
-----------------------------------------------------------------
-- Arquivo   : 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;
```

A seguir seguem os novos códigos.

A unidade de controle foi refatorada para facilitar a manutenção e compreensão da máquina de estados. Além disso houve uma mudança na representação dos estados no display de 7 segmentos: parte de cima do display agora mostra o processo de qual o estado faz parte (Inicial (I), Jogada (J), Gravação (G), Repete (R) e Reseta Memória (Z); e a parte de baixo a etapa do processo (1,2,3,...). Os estados que não pertencem necessariamente a um processo mantém seus nomes e símbolos originais (O,C,L,E e F). Como é ilustrado na figura a seguir:



Seguem os códigos da unidade de controle e do novo conversor para 7 segmentos, usado para os estados:

```
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;
    fimC:       in  std_logic;
    fimL:       in  std_logic;
    timeout: in std_logic;
    jogada:   in std_logic;
    enderecoIgualLimite: in std_logic;
    espera: out std_logic;
    zera:      out std_logic;
    conta_end:     out std_logic;
    conta_lim: out std_logic;
    conta_T: out std_logic;
    zera_T: out std_logic;
    zera_lim: out std_logic;
    pronto:    out std_logic;
    db_estado: out std_logic_vector(4 downto 0);
    acertou: out std_logic;
    errou: out std_logic;
    registra: out std_logic;
    igual: in std_logic;
    escreve: out std_logic;
    zera_2: out std_logic;
    reset_m: out std_logic;
    conta_2: out std_logic;
    timeout2: in std_logic;
   repete: in std_logic;
    chegou: in std_logic
  );
end entity;

architecture fsm of unidade_controle is
  type t_estado is (O, I1,I2,I3,I4,I5, J1,J2,J3 , C,  E,F,  L,
G1,G2,G3,G4 ,R1,R2,R3,R4 ,Z1,Z2,Z3);
  signal Eatual, Eprox: t_estado;
```

```
begin
  -- memoria de estado
  process (clock,reset)
  begin
    if reset='1' then
      Eatual <= O;
    elsif clock'event and clock = '1' then
      Eatual <= Eprox;
    end if;
  end process;

  -- logica de proximo estado
  Eprox <=
      O when  Eatual=O and iniciar='0' else
      I1 when  Eatual=O and iniciar='1' else
      I2 when Eatual=I1 else
      I2 when Eatual=I2 and jogada ='0' else
      I3 when Eatual=I2 and jogada ='1' else
      I4 when Eatual=I3 else
      I5 when Eatual=I4 else
      J1 when Eatual=I5 else
      J1 when Eatual=J1 and jogada='0' and timeout='0' else
      J2 when Eatual=J1 and jogada='1'and timeout='0' else
      C when Eatual=J2 else
      J3 when Eatual=C and enderecoIgualLimite='0' and igual='1'
else
      F when  Eatual=C and igual='0' else
      F when Eatual=J1 and timeout='1'else
      F when Eatual=G2 and timeout='1' else
    L when Eatual=G4 else
      G1 when Eatual=C and enderecoIgualLimite='1' and igual='1' and
fimC='0' else
    G2 when Eatual=G1 else
    G2 when Eatual=G2 and jogada ='0' and timeout='0' else
      G3 when Eatual=G2 and jogada ='1' and timeout='0' else
    G4 when Eatual=G3 else
      J1 when  Eatual=J3 else
      J1 when Eatual=L else
      E when  Eatual=C and FimC='1' and igual ='1' else
      Z1 when  Eatual=E and iniciar='1' else
```

```
      E when  Eatual=E and iniciar='0' else
      Z1 when  Eatual=F and iniciar='1' and repete='0' else
      F when  Eatual=F and iniciar='0' and repete='0'else
    Z2 when Eatual =Z1 else
    Z3 when Eatual=Z2 and fimC='0' else
    I1 when Eatual =Z2 and fimC='1' else
    Z2 when Eatual=Z3 else
      R1 when  Eatual=F and repete='1' else
      R2 when Eatual=R1 else
      R2 when Eatual = R2 and timeout2 ='0' and fimC='0' else
    F when Eatual=R2 and timeout2 ='0' and fimC='1' else
      R3 when Eatual =R2 and timeout2='1' else
      R4 when Eatual =R3 else
      R2 when Eatual=R4 and chegou='0' else
      F when Eatual=R4 and chegou='1' else
      O;

  -- logica de saída (maquina de Moore)
  with Eatual select
    registra <= '0' when O | I1 | C | J3 | E | F | J1,
                '1' when J2 | G3 | I3,
                '0' when others;

  with Eatual select
    zera_T <= '1' when G1 | L | J3 | I1 | I5,
                '0' when others;
  with Eatual select
    zera_2 <= '1' when  I1 | R4 | R1,
                '0' when others;

  with Eatual select
    reset_m <='1' when Z3 | Z2,
                '0' when others;

  with Eatual select
    conta_T <= '1' when J1 | G2,
                '0' when others;
with Eatual select
    conta_2 <= '1' when R2,
                '0' when others;
```

```
with Eatual select
  espera <= '1' when G2 | I2,
            '0' when others;

with Eatual select
  escreve <= '0' when O | I1 | C | J3 | E | F | J1,
             '1' when G4 | I4 | Z2,
             '0' when others;

             with Eatual select

  zera <=    '0' when O | C | J3 | E | F | J2 | J1,
             '1' when I1 | L | I5 | R1 | Z1,
             '0' when others;

with Eatual select
  conta_end <=  '0' when O | I1 | C | E | F | J2 | J1,
                '1' when J3 | G1 | R3 | Z3,
                '0' when others;

with Eatual select
    conta_lim <=  '0' when O | I1 | C | E | F | J2 | J1 | J3,
              '1' when L,
              '0' when others;

with Eatual select
    zera_lim <=  '0' when O | C | E | F | J2 | J1 | J3 | L,
                 '1' when I1,
                 '0' when others;

with Eatual select
  pronto <= '0' when O | I1 | C | J3 | J2 | J1,
            '1' when E | F | R1 | R2 | R3 | R4,
            '0' when others;

with Eatual select
  acertou <='0' when O | I1 | C | J3 | F | J2 | J1,
            '1' when E,
            '0' when others;
```

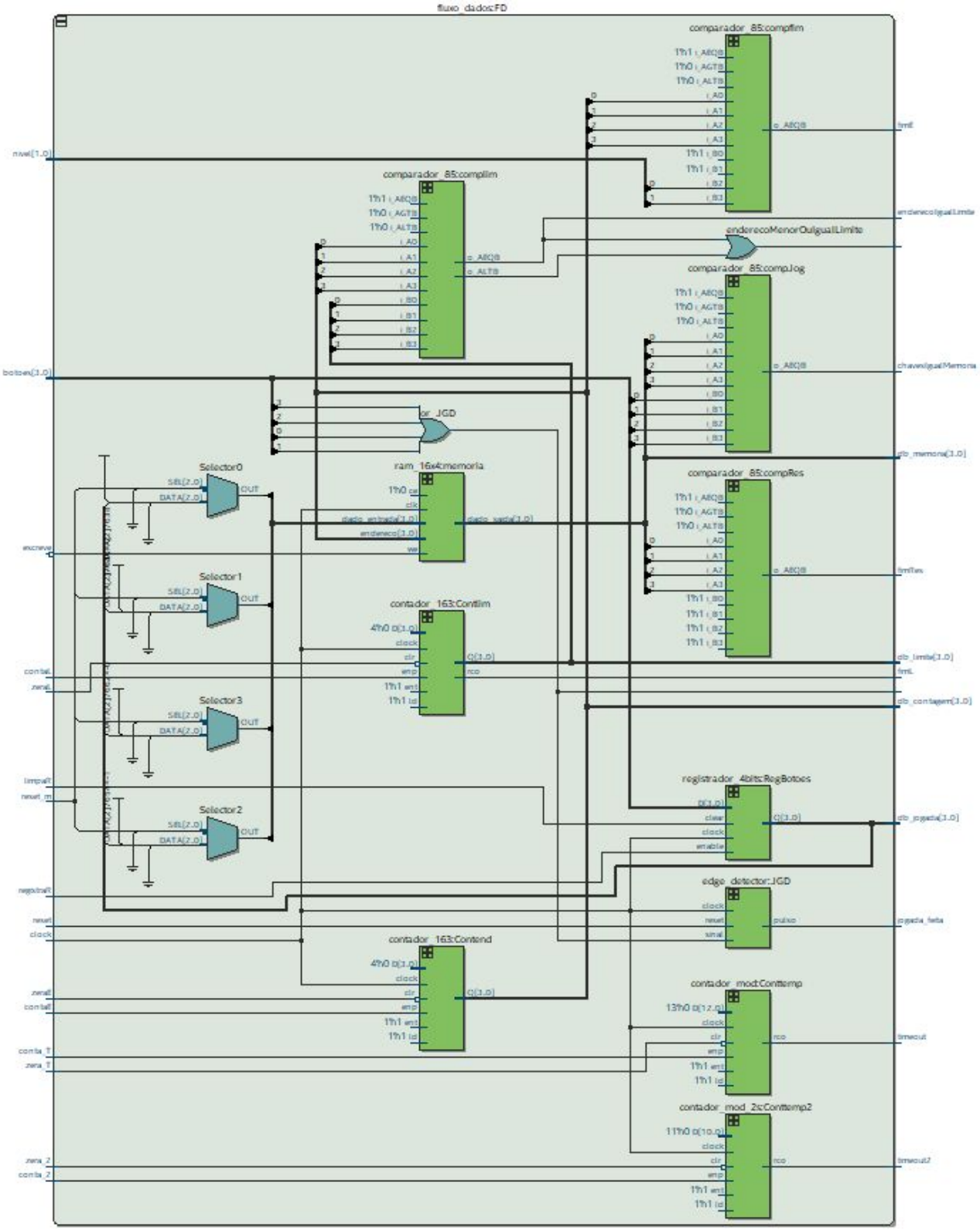
```
with Eatual select
  errou <=  '0' when O | I1 | C | J3 | E | J2 | J1,
            '1' when F | R1 | R2 | R3 | R4,
            '0' when others;

-- saida de depuracao (db_estado)
with Eatual select
  db_estado <= "00000" when O,

               "00001" when I1,
               "00010" when I2,
               "00011" when I3,
               "00100" when I4,
               "00101" when I5,

               "00110" when J1,
               "00111" when J2,
               "01000" when J3,

               "01001" when Z1,
               "01010" when Z2,
               "01011" when Z3,

               "01100" when C,
               "01101" when L,
               "01110" when E,
               "01111" when F,

               "10000" when G1,
               "10001" when G2,
               "10010" when G3,
               "10011" when G4,

               "10100" when R1,
               "10101" when R2,
               "10110" when R3,
               "10111" when R4,
               "11111" when others;
```

```
end fsm;
```

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

entity estado7seg is
    port (
        estado : in  std_logic_vector(4 downto 0);
        display : out std_logic_vector(6 downto 0)
    );
end estado7seg;

architecture comportamental of estado7seg is
begin

  display <= "1000000"  when estado="00000" else -- O

  "1101111" when estado="00001"  else -- I1
  "1110111" when estado="00010"  else -- I2
  "1111011" when estado= "00011"  else -- I3
  "0111111" when estado="00100"  else -- I4
  "0100011" when estado="00101"  else -- I5

  "1001111" when estado="00110"  else -- J1
  "1010111" when estado="00111"  else -- J2
  "1011011" when estado="01000"  else -- J3

  "1001100" when estado="01001"  else -- Z1
  "1010100" when estado="01010"  else -- Z2
  "1011000" when estado="01011"  else -- Z3

  "1000110" when estado="01100"  else -- C
  "1000111" when estado="01101"  else --L
  "0000110" when estado="01110"  else --E
  "0001110" when estado="01111"  else --F

  "1101110" when estado="10000"  else --G1
  "1110110" when estado="10001"  else -- G2
```

```
  "1111010" when estado="10010"  else --G3
  "0111110" when estado="10011"  else --G4

 "1101101" when estado="10100"  else --R1
 "1110101" when estado="10101"  else --R2
 "1111001" when estado="10110"  else --R3
 "0111101" when estado="10111" else
 "0000000";

end comportamental;
```

A seguir segue o diagrama de blocos do fluxo de dados modificado, com o novo comparador e o multiplexador para escolha do “final”:



A seguir segue o código VHDL do fluxo de dados:

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

```
entity fluxo_dados is
    port (
        clock : in std_logic;
      reset: in std_logic;
      zera_T: in std_logic;
      conta_T: in std_logic;
      timeout: out std_logic;
        zeraE : in std_logic;
        limpaR: in std_logic;
        registraR: in std_logic;
        zeraL: in std_logic;
        contaL: in std_logic;
        contaE : in std_logic;
        escreve: in std_logic;
        botoes : in std_logic_vector (3 downto 0);
        fimE: out std_logic;
        fimL: 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);
        db_limite: out std_logic_vector (3 downto 0);
        jogada_feita: out std_logic;
        db_jogada: out std_logic_vector (3 downto 0);
        chavesIgualMemoria: out std_logic;
        enderecoMenorOuIgualLimite: out std_logic;
        enderecoIgualLimite: out std_logic;
        reset_m: in std_logic;
        zera_2: in std_logic;
        conta_2: in std_logic;
        timeout2: out std_logic;
      fimRes: out std_logic;
      nivel: in std_logic_vector (1 downto 0)
    );
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 contador_mod is
      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 (12 downto 0);
        Q : out std_logic_vector (12 downto 0); -- sinais de saída
        rco : out std_logic
 );
  end component;

  component contador_mod_2s is
    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 (10 downto 0);
    Q : out std_logic_vector (10 downto 0); -- sinais de saída
    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,zeraE_baixo,igual_out,menor,great_o,menor_o,enable_cin,rco_out
c,or_JGD,zeraL_baixo,rco_outL,fim_L: std_logic;
    signal enderecoMenorqueLimite, IgualLimite,
ZeraT_baixo,zera2_baixo: std_logic;
    signal s_jogada, entrada_m: std_logic_vector (3 downto 0);
    signal contador_out, s_dado,s_endereco,s_lim: std_logic_vector
(3 downto 0);
    signal final: std_logic_vector (3 downto 0);
    begin


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

        ZeraL_baixo<= not ZeraL;
        Contlim: contador_163 port map (
            clock => clock,
            clr => ZeraL_baixo,
```

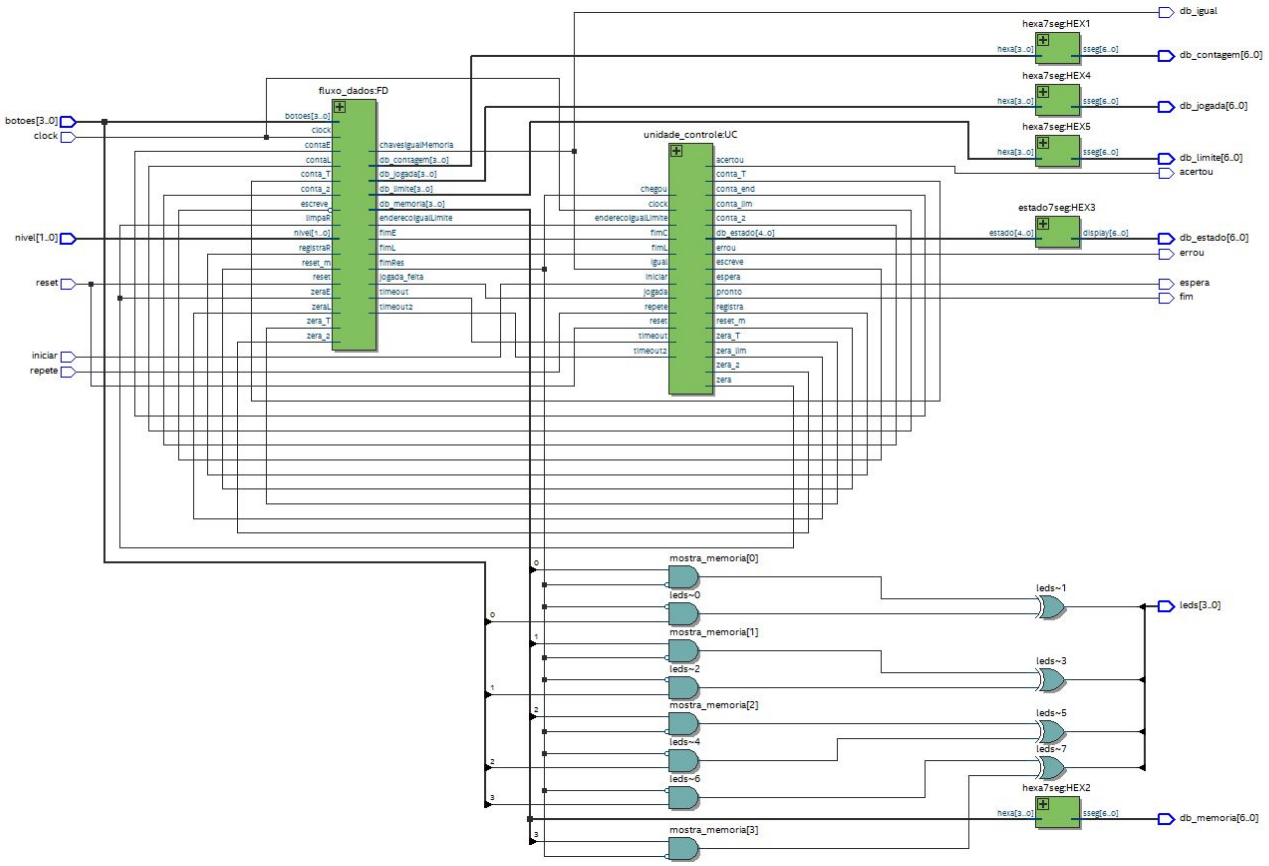
```
          ld    => '1',
          ent   => '1',
          enp   =>contaL,
          D     => "0000",
          Q     => s_lim,
          rco   => rco_outL
      );
      fimL<=rco_outL;
      db_limite<=s_lim;
    enable_Cin<=contaE;
    db_contagem<=s_endereco;

    ZeraT_baixo<= not Zera_T;

    Conttemp: contador_mod port map (
      clock => clock,
      clr => ZeraT_baixo,
      ld    => '1',
      ent   => '1',
      enp=>conta_T,
      D=> "0000000000000",
      rco=>timeout
  );
  Zera2_baixo<= not Zera_2;
  Conttemp2: contador_mod_2s port map (
    clock => clock,
    clr => Zera2_baixo,
    ld    => '1',
    ent   => '1',
    enp=>conta_2,
    D=> "00000000000",
    rco=>timeout2
);

    complim: comparador_85 port map (
      i_A3  =>s_endereco(3),
        i_B3   => s_lim(3),
        i_A2   =>s_endereco(2),
        i_B2   => s_lim(2),
        i_A1   =>s_endereco(1),
```

```
          i_B1   => s_lim(1),
          i_A0   =>s_endereco(0),
          i_B0   => s_lim(0),
          i_AGTB =>'0',
          i_ALTB => '0',
          i_AEQB => '1',
          -- saidas
          o_ALTB =>enderecoMenorQueLimite,
          o_AEQB =>IgualLimite
    );

    compfim: comparador_85 port map (
        i_A3  =>s_endereco(3),
          i_B3   => final(3),
          i_A2   =>s_endereco(2),
          i_B2   => final(2),
          i_A1   =>s_endereco(1),
          i_B1   => final(1),
          i_A0   =>s_endereco(0),
          i_B0   => final(0),
          i_AGTB =>'0',
          i_ALTB => '0',
          i_AEQB => '1',
          -- saidas
          o_AEQB =>FimE
    );

    with nivel select
    final<="0011" when "00",
          "0111" when "01",
          "1011" when "10",
          "1111" when "11",
          "0000" when others;


    enderecoIgualLimite<=IgualLimite;
    enderecoMenorOuIgualLimite<= enderecoMenorQueLimite or
IgualLimite;

        compJog: comparador_85 port map (
```

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

    compRes: comparador_85 port map (
        i_A3  =>s_dado(3),
        i_B3   => '1',
        i_A2   =>s_dado(2),
        i_B2   => '1',
        i_A1   =>s_dado(1),
        i_B1   => '1',
        i_A0   =>s_dado(0),
        i_B0   => '1',
        i_AGTB =>'0',
        i_ALTB => '0',
        i_AEQB => '1',
        -- saidas
        o_AEQB => fimRes
    );


    memoria: ram_16x4 port map(
      clk=> clock,
        endereco => s_endereco,
        dado_entrada => entrada_m,
        we => escreve,
        ce => '0',
```

```
            dado_saida => s_dado
        );
      with reset_m select
      entrada_m<=s_jogada when '0',
      "1111"  when '1',
      "0000" when others;
        db_memoria<=s_dado;
        chavesIgualMemoria<=igual_out;

        RegBotoes: registrador_4bits port map(
        clock=>clock,
        clear=>limpaR,
        enable=>registraR,
        D=>botoes,
        Q=>s_jogada);
        db_jogada<=s_jogada;
        or_JGD<=botoes(0) or botoes(1) or botoes(2) or botoes(3);
        JGD: edge_detector port map(
            clock=>clock,
            reset=>reset,
            sinal=>or_JGD,
            pulso=>jogada_feita
        );
        db_tem_jogada<=or_JGD;
    end architecture;
```

Por fim, segue o diagrama RTL do circuito no geral:



E o código VHDL do circuito:

```
library ieee;
use ieee.std_logic_1164.all;
entity circuito_semana1 is
    port(
        clock : in std_logic;
        reset : in std_logic;
        iniciar : in std_logic;
        botoes : in std_logic_vector (3 downto 0);
        acertou : out std_logic;
        errou : out std_logic;
        fim : out std_logic;
        espera: out std_logic;
        leds: out std_logic_vector (3 downto 0);
        repete: in std_logic;
        nivel: in std_logic_vector (1 downto 0);
        db_limite : out std_logic_vector (6 downto 0);
        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)
    );
end entity;

architecture estrutural of circuito_semana1 is
    component fluxo_dados is
        port(
            clock : in std_logic;
          reset: in std_logic;
      zera_T: in std_logic;
      conta_T: in std_logic;
      timeout: out std_logic;
        zeraE : in std_logic;
        limpaR: in std_logic;
        registraR: in std_logic;
        zeraL: in std_logic;
        contaL: in std_logic;
        contaE : in std_logic;
        escreve: in std_logic;
        botoes : in std_logic_vector (3 downto 0);
        fimE: out std_logic;
        fimL: 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);
        db_limite: out std_logic_vector (3 downto 0);
        jogada_feita: out std_logic;
        db_jogada: out std_logic_vector (3 downto 0);
        chavesIgualMemoria: out std_logic;
        enderecoMenorOuIgualLimite: out std_logic;
        enderecoIgualLimite: out std_logic;
        reset_m: in std_logic;
        zera_2: in std_logic;
        conta_2: in std_logic;
        timeout2: out std_logic;
          fimRes:out std_logic;
          nivel: in std_logic_vector (1 downto 0)
```

```
    );
end component;

component unidade_controle is
    port(
        clock:     in  std_logic;
reset:     in  std_logic;
iniciar:   in  std_logic;
fimC:       in  std_logic;
fimL:       in  std_logic;
timeout: in std_logic;
jogada:   in std_logic;
enderecoIgualLimite: in std_logic;
espera: out std_logic;
zera:      out std_logic;
conta_end:     out std_logic;
conta_lim: out std_logic;
conta_T: out std_logic;
zera_T: out std_logic;
zera_lim: out std_logic;
pronto:    out std_logic;
db_estado: out std_logic_vector(4 downto 0);
acertou: out std_logic;
errou: out std_logic;
registra: out std_logic;
igual: in std_logic;
escreve: out std_logic;
zera_2: out std_logic;
reset_m: out std_logic;
    conta_2: out std_logic;
    timeout2: in std_logic;
    repete: in std_logic;
    chegou: in std_logic
    );
end component;

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

```
        );
        end component;

    component estado7seg is
        port (
                estado : in  std_logic_vector(4 downto 0);
                display : out std_logic_vector(6 downto 0)
        );
        end component;

    signal conta4,memo4,joga4,lim4,botoes_led: std_logic_vector (3
downto 0);
    signal
conta,zeraE,registra,clk,jogada,igual_i,escreve_baixo,escreve,db_tem
_jogada,db_clock: std_logic;
    signal
zeraL,contaL,contaE,fimE,fimL,jogada_feita,chavesIgualMemoria,endere
coIgualLimite_i,zera_T,conta_T,timeout: std_logic;
    signal zera_2,
conta_2,timeout2,reset_m,tudo_aceso_baixo,tudo_aceso,chegou:
std_logic;
    signal mostra_memoria: std_logic_vector (3 downto 0);
    signal estad4: std_logic_vector (4 downto 0);
    begin
        tudo_aceso_baixo<= not chegou;
        tudo_aceso<= not tudo_aceso_baixo;
        mostra_memoria(0)<= memo4(0) and tudo_aceso_baixo;
        mostra_memoria(1)<= memo4(1) and tudo_aceso_baixo;
        mostra_memoria(2)<= memo4(2) and tudo_aceso_baixo;
        mostra_memoria(3)<= memo4(3) and tudo_aceso_baixo;
        leds(0)<=(botoes(0) and tudo_aceso_baixo) xor
mostra_memoria(0);
        leds(1)<=(botoes(1) and tudo_aceso_baixo) xor
mostra_memoria(1);
        leds(2)<=(botoes(2) and tudo_aceso_baixo) xor
mostra_memoria(2);
        leds(3)<=(botoes(3) and tudo_aceso_baixo) xor
mostra_memoria(3);
        clk<=clock;
        FD: fluxo_dados port map(
```

```
            clock =>clk,
                reset=> reset,
        zeraE =>zeraE,
        limpaR=> zeraE,
        registraR=>registra,
        zeraL=>zeraL,
        contaL=>contaL,
        contaE =>contaE,
        escreve=>escreve_baixo,
        botoes =>botoes,
        fimE=>fimE,
        fimL=>fimL,
        db_tem_jogada=>db_tem_jogada,
        db_contagem =>conta4,
        db_memoria=>memo4,
        db_limite=>lim4,
        jogada_feita=>jogada_feita,
        db_jogada=>joga4,
        chavesIgualMemoria=>chavesIgualMemoria,
        enderecoIgualLimite=>enderecoIgualLimite_i,
        zera_T=>zera_T,
        conta_T=>conta_T,
        timeout=>timeout,
        zera_2=>zera_2,
    reset_m=>reset_m,
        conta_2=>conta_2,
        timeout2=>timeout2,
          fimRes=>chegou,
          nivel=>nivel
        );
        db_igual<=chavesIgualMemoria;
        UC: unidade_controle port map(
            clock=>clock,
        reset=>reset,
        iniciar=>iniciar,
        fimC=>fimE,
        fimL=>fimL,
        jogada=>jogada_feita,
        enderecoIgualLimite=>enderecoIgualLimite_i,
        zera=>zeraE,
```

```
        conta_end=>contaE,
        conta_lim=>contaL,
        zera_lim=>zeraL,
        pronto=>fim,
        db_estado=>estad4,
        acertou=>acertou,
        errou=>errou,
        registra=>registra,
        igual=>chavesIgualMemoria,
        escreve=>escreve,
        espera=>espera,
        zera_T=>zera_T,
        conta_T=>conta_T,
        timeout=>timeout,
        zera_2=>zera_2,
    reset_m=>reset_m,
        conta_2=>conta_2,
        timeout2=>timeout2,
        repete=> repete,
        chegou=>chegou
        );

        escreve_baixo<= not escreve;
        db_clock<=clk;
        HEX1: hexa7seg port map(
            hexa =>conta4,
            sseg =>db_contagem
        );

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

        HEX3: estado7seg port map(
            estado =>estad4,
            display =>db_estado
        );

        HEX4: hexa7seg port map(
```

```
            hexa=>joga4,
            sseg=> db_jogada
        );
        HEX5: hexa7seg port map(
            hexa=>lim4,
            sseg=> db_limite
        );
end architecture;
```

2.2) Plano de testes

Cenário #1 – Acerto de todas as jogadas no nivel 00				
#	**Operação**	**Sinais de Entrada**	**Resultado Esperado**	**Resultado Observado**
c.i.	**Condições Iniciais**			
1	**Iniciar ciclo**	**iniciar = 1, nivel = 00**		
2	**Acertar jogada**	**botoes = 0001**	**igual = 1 db_limite = 1**	**igual = 1 db_limite = 1**
3	**Acertar jogada**	**botoes = 0001, botoes = 0010**	**igual = 1 db_limite = 2**	**igual = 1 db_limite = 2**
4	**Acertar jogada**	**botoes = 0001, botoes = 0010, botoes = 0100**	**igual = 1 db_limite = 3**	**igual = 1 db_limite = 3**
5	**Acertar jogada**	**botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000**	**igual = 1 db_limite = 4 fim = 1 acertou = 1**	**igual = 1 db_limite = 4 fim = 1 acertou = 1**

Cenário #2 – Acerto de todas as jogadas no nivel 01				
#	**Operação**	**Sinais de Entrada**	**Resultado Esperado**	**Resultado Observado**
c.i.	**Condições Iniciais**			
1	**Iniciar ciclo**	**iniciar = 1, nivel = 01**		
2	**Acertar jogada**	**botoes = 0001**	**igual = 1 db_limite = 1**	**igual = 1 db_limite = 1**

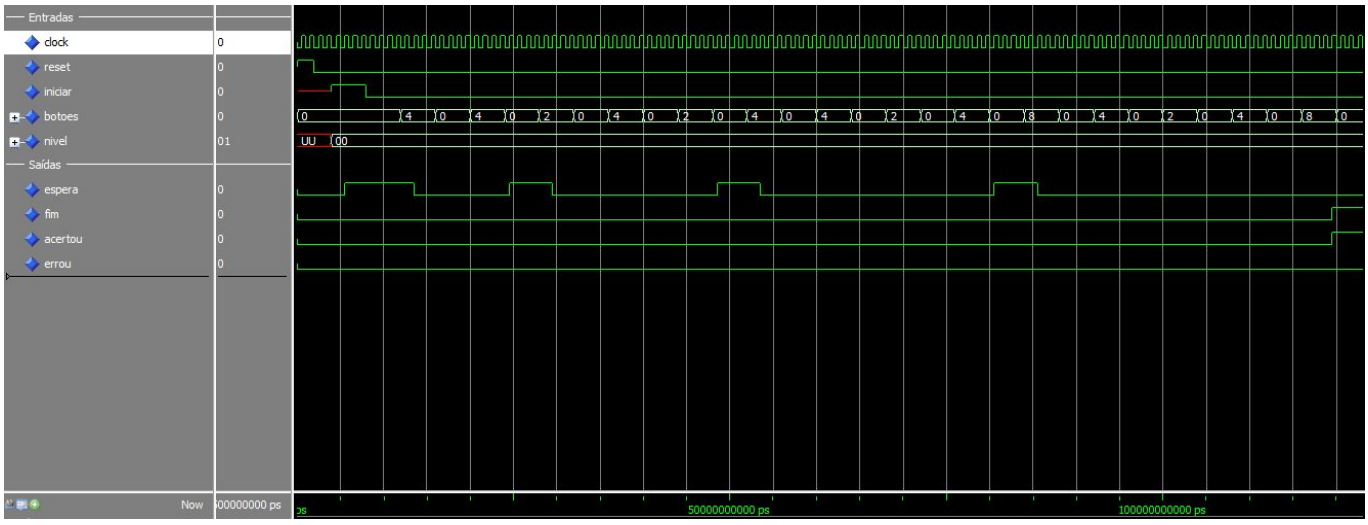
3	**Acertar jogada**	**botoes = 0001, botoes = 0010**	**igual = 1 db_limite = 2**	**igual = 1 db_limite = 2**
4	**Acertar jogada**	**botoes = 0001, botoes = 0010, botoes = 0100**	**igual = 1 db_limite = 3**	**igual = 1 db_limite = 3**
5	**Acertar jogada**	**botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000**	**igual = 1 db_limite = 4**	**igual = 1 db_limite = 4**
6	**Acertar jogada**	**botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100**	**igual = 1 db_limite = 5**	**igual = 1 db_limite = 5**
7	**Acertar jogada**	**botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010**	**igual = 1 db_limite = 6**	**igual = 1 db_limite = 6**
8	**Acertar jogada**	**botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001**	**igual = 1 db_limite = 7**	**igual = 1 db_limite = 7**
9	**Acertar jogada**	**botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001, botoes = 0001**	**igual = 1 db_limite = 8 fim = 1 acertou = 1**	**igual = 1 db_limite = 8 fim = 1 acertou = 1**

Cenário #3 – Erro na 4ª jogada				
#	**Operação**	**Sinais de Entrada**	**Resultado Esperado**	**Resultado Observado**
c.i.	**Condições Iniciais**			
1	**Iniciar Ciclo**	**iniciar = 1, nivel = 10**		
2	**Acertar jogadas**	**botoes = 0001**	**igual = 1 db_limite = 1**	**igual = 1 db_limite = 1**
3		**botoes = 0001, botoes = 0010**	**igual = 1 db_limite = 2**	**igual = 1 db_limite = 2**
4		**botoes = 0001, botoes = 0010, botoes = 0100**	**igual = 1 db_limite = 3**	**igual = 1 db_limite = 3**
5	**Errar jogada**	**botoes = 0010**	**fim = 1, errou =1**	**fim = 1, errou =1**

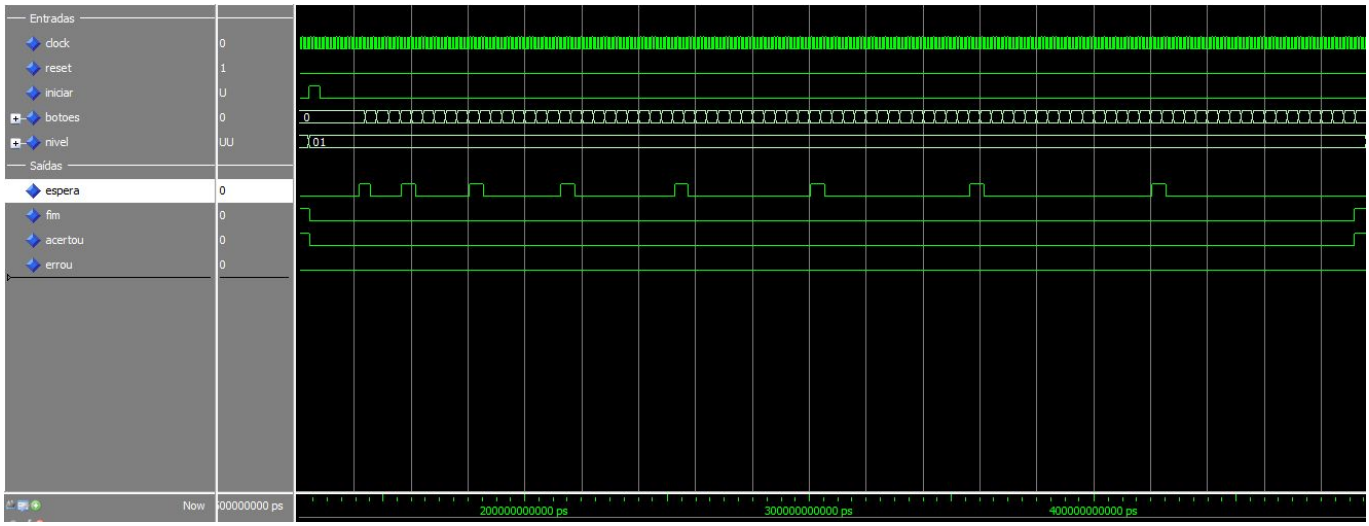
Cenário #4 – Perder por timeout após 3ª jogada				
#	**Operação**	**Sinais de Entrada**	**Resultado Esperado**	**Resultado Observado**
c.i.	**Condições Iniciais**			
1	**Iniciar Ciclo**	**iniciar = 1, nivel = 11**		
2	**Acertar jogadas**	**botoes = 0001**	**igual = 1 db_limite = 1**	**igual = 1 db_limite = 1**
3		**botoes = 0001, botoes = 0010**	**igual = 1 db_limite = 2**	**igual = 1 db_limite = 2**
4		**botoes = 0001, botoes = 0010, botoes = 0100**	**igual = 1 db_limite = 3**	**igual = 1 db_limite = 3**
5	**Esperar 5 segundos**		**fim = 1, errou =1**	**fim = 1, errou =1**

2.3) Simulação do circuito

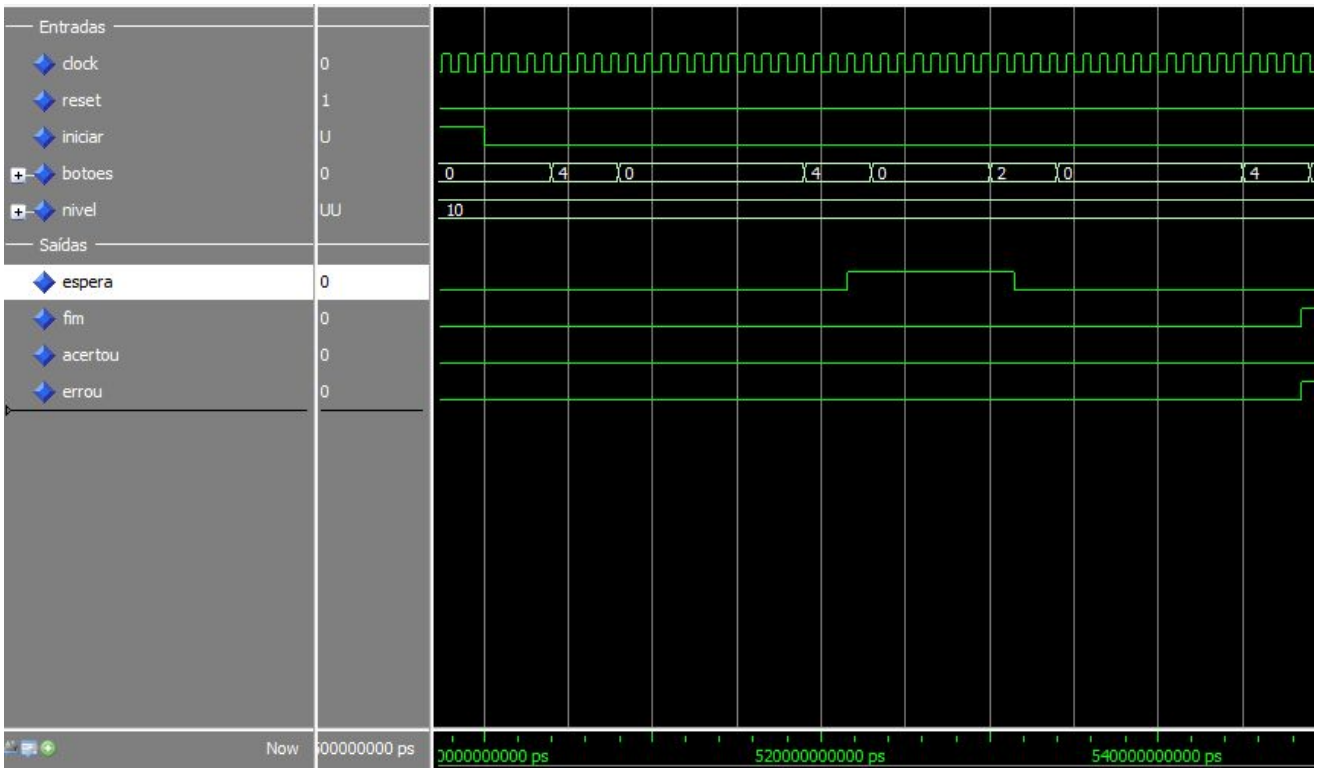
2.3.1) Cenário #1



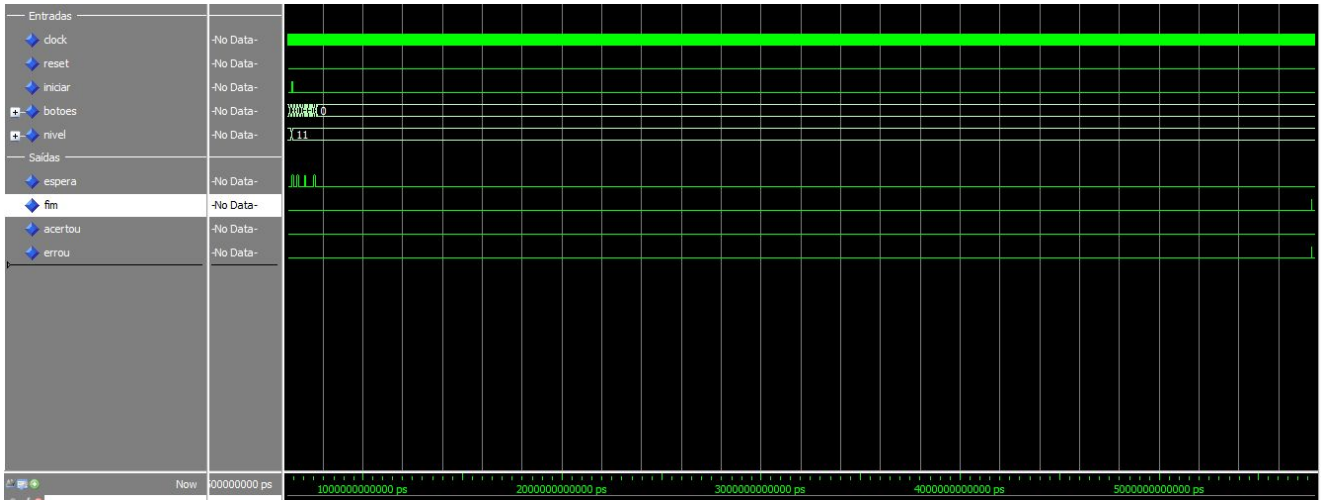
2.3.2) Cenário #2



2.3.3) Cenário #3



2.3.4) Cenário #4



2.4) Designação de pinos

Tabela 1: Designação de Pinos para a Atividade 2

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**
JOGAR	**GPIO_0_D17**	PIN_P19	**StaticIO – Button 0/1 – DIO2**
BOTOES(0)	**GPIO_0_D19**	PIN_P17	**StaticIO – Button 0/1 – DIO3**
BOTOES(1)	**GPIO_0_D21**	PIN_M18	**StaticIO – Button 0/1 – DIO4**
BOTOES(2)	**GPIO_0_D23**	PIN_L17	**StaticIO – Button 0/1 – DIO5**
BOTOES(3)	**GPIO_0_D25**	PIN_K17	**StaticIO – Button 0/1 – DIO6**
REPETIR	**GPIO_0_D27**	PIN_P18	**StaticIO – Button 0/1 – DIO7**
NIVEL(0)	**GPIO_0_D29**	PIN_R17	**StaticIO – Switch Push/Pull – DIO8**
NIVEL(1)	**GPIO_0_D31**	PIN_T20	**StaticIO – Switch Push/Pull – DIO9**
LEDS(0)	**LED LEDR0**	PIN_AA2	-
LEDS(1)	**LED LEDR1**	PIN_AA1	-
LEDS(2)	**LED LEDR2**	PIN_W2	-
LEDS(3)	**LED LEDR3**	PIN_Y3	-
ESPERA	**LED LEDR6**	PIN_U2	-
PERDEU	**LED LEDR7**	PIN_U1	-

GANHOU	**LED LEDR8**	PIN_L2	**-**
FIM	**LED LEDR9**	PIN_L1	**-**
db_jogadafinal	**LED LEDR4**	PIN_N2	**-**
db_timeout	**LED LEDR5**	PIN_N1	
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_limite	**Display HEX3**	[0] PIN_Y16 [1] PIN_W16 [2] PIN_Y17 [3] PIN_V16 [4] PIN_U17 [5] PIN_V18 [6] PIN_V19	**-**
db_nivel	**Display HEX4**	[0] PIN_U20 [1] PIN_Y20 [2] PIN_V20 [3] PIN_U16 [4] PIN_U15 [5] PIN_Y15 [6] PIN_P9	**-**
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	**-**