



**Laboratório Digital I - PCS3635**

**Relatório da Experiência 7:**

**Projeto do Jogo do Desafio da Memória**

Marco Aurélio C. O. Prado - NUSP 11257605

Victor Hoefling Padula - NUSP 10770051

Turma 04 - Bancada A1

São Paulo - SP

05/03/2021

## **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\_exp7.vhd”**

### **2.1) Descrição do funcionamento do circuito**

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.

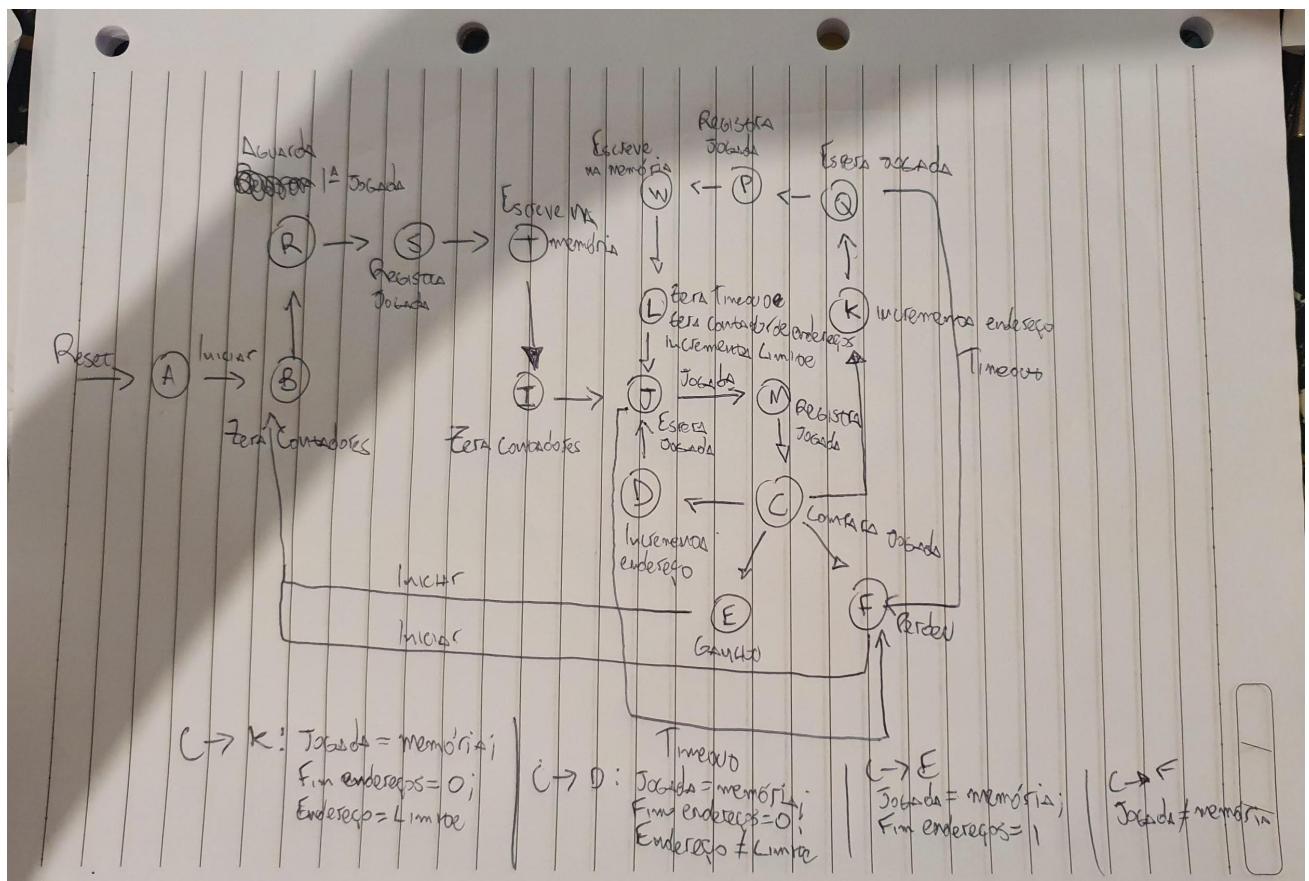
Para iniciar o jogo, deve-se apertar o botão “iniciar”. Após isso, o circuito aguardará pela primeira jogada, que deve ser inserida pelo jogador através dos botões ligados aos sinais “botoes”. Após isso, o jogador deve repetir a primeira jogada e acrescentar mais uma. Na rodada seguinte, o jogador deve 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.

Para implementação do timeout foi acrescentado ao fluxo de dados um contador\_163 modificado, chamado contador\_mod. Ele tem funcionamento idêntico ao 163, com a diferença de que ele conta até 5000 ao invés de 15.

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 B, onde reinicia todos os contadores. Depois, vai para o estado R, onde aciona o led Espera e aguarda a primeira jogada a ser gravada. Então segue para os estados de lógica de gravação, S, onde registra a jogada, T, onde guarda a jogada na memória, e depois I, onde zera o contador de endereços e do timeout. Depois, entra no loop do jogo, indo para o estado J, 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 M, se o tempo acabar ele vai para o estado “perdeu”, ou F. No estado M, 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 D; se o endereço é igual ao limite mas o contador de endereços não está no fim, ele vai para o estado K; se ele acertou e estiver no último endereço da memória ele vai

para o estado E (ganhou). No estado D ele incrementa o contador de endereços, zera o timeout e volta para o estado J. no estado K ele também incrementa o contador de endereços e zera o timeout, mas vai para o estado Q. No estado Q ele aguarda a próxima jogada, aciona o timeout de forma similar ao J, mas se o jogador fizer uma jogada a tempo ele vai para os estados de lógica de gravação, P, onde registra a jogada, W, 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 B.

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



## 2.2) Descrição VHDL do circuito “circuito\_exp7.vhd”

Descrição dos componentes contador\_163, contador\_mod, comparador\_85, ram16x4, edge\_detector, hex7seq e registrador\_4bits, 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 saida  

        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

```

Vale notar que esse contador vai até 5000

```

-----  

-- 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 compararam A e B sem levar em
    -- consideracao as entradas de cascamenteo
    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, sao 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 : modulo de memoria RAM sincrona 16x4
--             sinais we e ce ativos em baixo
--             codigo ADAPTADO do codigo 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",
                                         "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 ativação de we (ativo baixo)
            if (we = '0')
                then memoria(to_integer(unsigned(endereco))) <=
dado_entrada;
            end if;

            end if;
        end if;
    end process;

    -- saída da memória
    dado_saida <= memoria(to_integer(unsigned(endereco)));
end architecture ram_modelsim;

```

Vale notar que todos os endereços da ram guardam o valor 1111, que serão substituídos pelas jogadas gravadas

```

-----
-----
-- Arquivo   : edge_detector.vhd
-- Projeto   : Experiencia 05 - Consideracoes de Projeto com FPGA
-----
-----
-- Descricao : detector de borda
--             gera um pulso na saída de 1 período de clock
--             a partir da detecção da borda de subida da entrada
--             sinal de reset ativo em alto
--             > adaptado a partir de código VHDL disponível em
--
https://surf-vhdl.com/how-to-design-a-good-edge-detector/
-----
-----
-- Revisões  :
--      Data          Versão  Autor           Descrição
--      29/01/2020    1.0     Edson Midorikawa  criação

```

```

-----
-----  

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;

```

```

-----
-- 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;

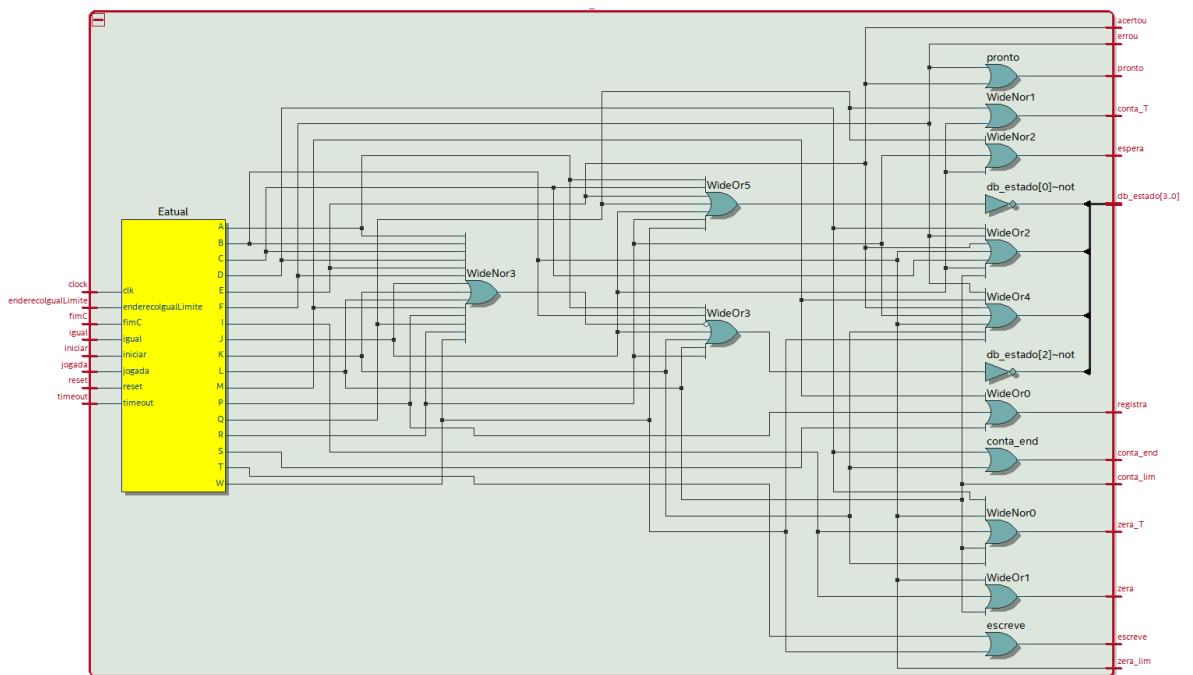
entity registrador_4bits is
port (
    clock:  in std_logic;
    clear:  in std_logic;
    enable: in std_logic;
    D:       in std_logic_vector(3 downto 0);
    Q:       out std_logic_vector(3 downto 0)
);
end entity;

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;
```

A seguir, segue a descrição da unidade de controle, de acordo com o diagrama:



```
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(3 downto 0);
acertou: out std_logic;
errou: out std_logic;
registra: out std_logic;
igual: in std_logic;
escreve: out std_logic
);
end entity;

architecture fsm of unidade_controle is
type t_estado is (A, B, C, D, E, F, M,J,L,K,W,Q,P,R,S,T,I);
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
    R when Eatual=B else
    R when Eatual=R and jogada ='0' else
    S when Eatual=R and jogada ='1' else
    T when Eatual=S else

```

```

I when Eatual=T else
J when Eatual=I else
J when Eatual=J and jogada='0' and timeout='0' else
M when Eatual=J and jogada='1' and timeout='0' else
C when Eatual=M else
D when Eatual=C and enderecoIgualLimite='0' and igual='1' else
F when Eatual=C and igual='0' else
F when Eatual=J and timeout='1' else
F when Eatual=Q and timeout='1' else
L when Eatual=W else
K when Eatual=C and enderecoIgualLimite='1' and igual='1' and
fimC='0' else
Q when Eatual=K else
Q when Eatual=Q and jogada ='0' and timeout='0' else
P when Eatual=Q and jogada ='1' and timeout='0' else
W when Eatual=P else
J when Eatual=D else
J when Eatual=L else
E when Eatual=C and FimC='1' and igual ='1' 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 | P | S,
      '0' when others;

with Eatual select
zera_T <= '1' when K | L | D | B | I,
      '0' when others;

with Eatual select
conta_T <= '1' when J | Q,
      '0' when others;

with Eatual select

```

```

espera <= '1' when Q | R,
          '0' when others;

with Eatual select
  escreve <= '0' when A | B | C | D | E | F | J,
            '1' when W | T,
            '0' when others;

          with Eatual select

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

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

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

with Eatual select
  zera_lim <=  '0' when A | C | E | F | M | J | D | L,
            '1' when B,
            '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

```

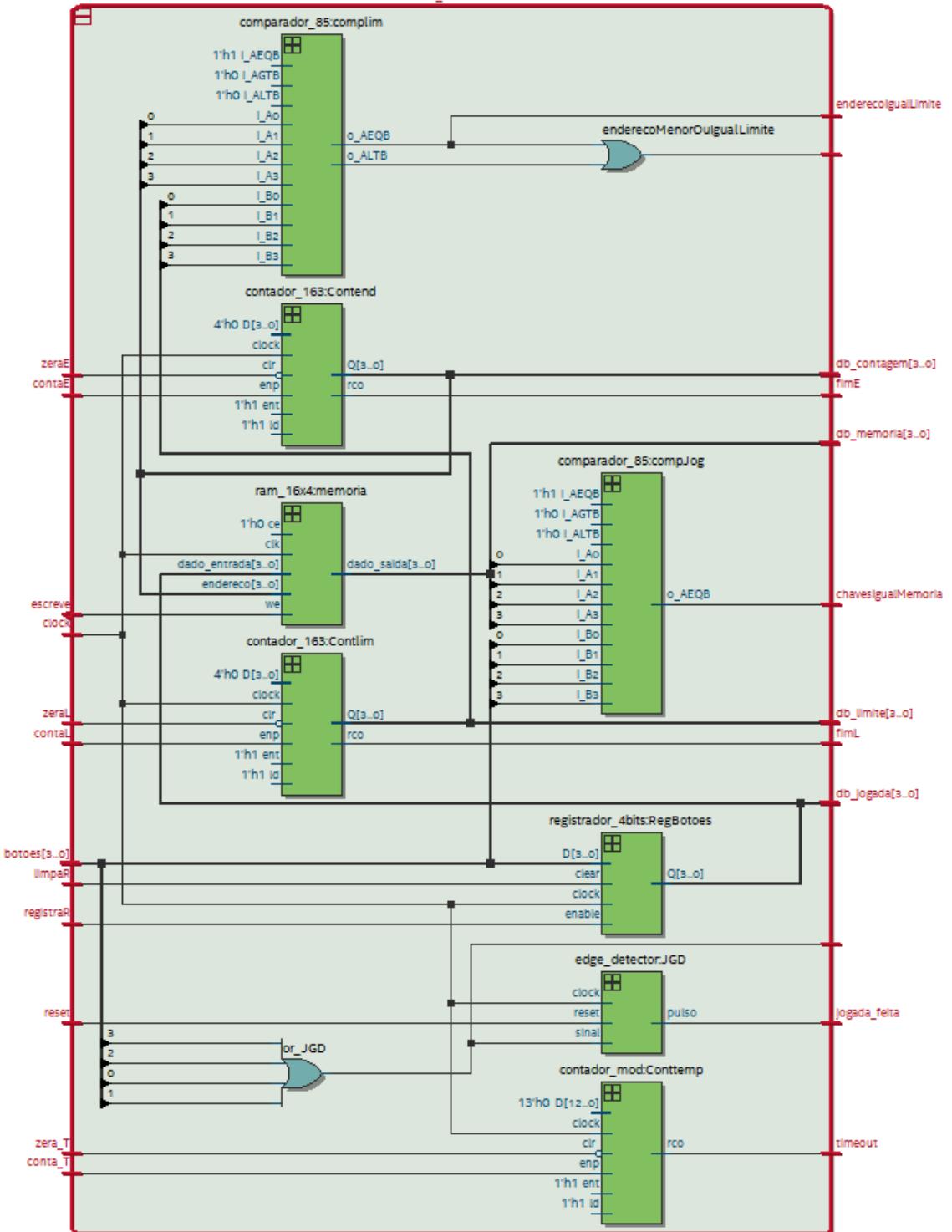
```

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
              "0100" when Q, --Q,4
              "0010" when R, --R,2
              "0011" when K, --K,3
              "0101" when P, -- P,5
              "0110" when W, --W,6
              "0111" when M,-- M,7
              "1000" when J, -- J,8
              "1001" when L, --L,9
              "1111" when F,          -- F
              "0001" when others;    -- 1
end fsm;

```

Depois, o fluxo de dados, que segue o projeto do diagrama:



```

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
);
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 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_EQB : in std_logic;
    -- saídas
    o_AGTB : out std_logic;
    o_ALTB : out std_logic;
    o_EQB : 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:
std_logic;
signal s_jogada: std_logic_vector (3 downto 0);
signal contador_out, s_dado,s_endereco,s_lim: 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
);
fimE<=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=> "0000000000000000",
    rco=>timeout

```

```

) ;

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_EQB  => '1',
    -- saidas
    o_ALTB =>enderecoMenorQueLimite,
    o_EQB  =>IgualLimite
);
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_EQB  => '1',
    -- saidas
    o_AGTB =>great,
    o_ALTB =>menor,
    o_EQB  => igual_out
);

```

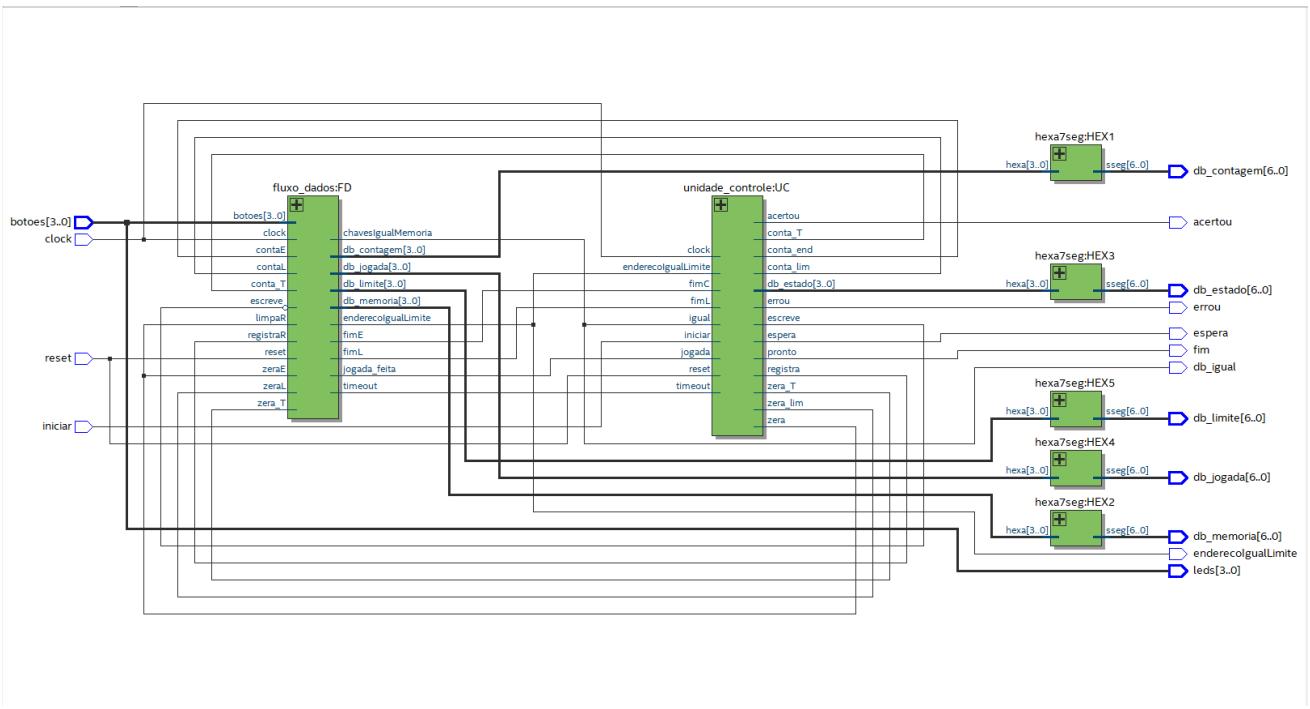
```

memoria: ram_16x4 port map(
    clk=> clock,
    endereco => s_endereco,
    dado_entrada => s_jogada,
    we => escreve,
    ce => '0',
    dado_saida => s_dado
);
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;

```

E por fim, o circuito geral:



```

library ieee;
use ieee.std_logic_1164.all;
entity circuito_exp7 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);
        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_exp7 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
    );
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(3 downto 0);
        acertou:    out std_logic;
        errou:      out std_logic;
        registra:   out std_logic;
        igual:      in std_logic;
        escreve:    out 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;

signal conta4,memo4,estad4,joga4,lim4,botoes_led:
std_logic_vector (3 downto 0);
signal
conta,zeraE,registra,clk,jogada,igual_i,escreve_baixo,escreve,db_temp_jogada,db_clock: std_logic;
signal
zeraL,contaL,contaE,fimE,fimL,jogada_feita,chavesIgualMemoria,enderecoIgualLimite_i,zera_T,conta_T,timeout: std_logic;
begin
    leds(0)<=botoes(0);
    leds(1)<=botoes(1);
    leds(2)<=botoes(2);
    leds(3)<=botoes(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
);
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
    ) ;

        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: hexa7seg port map(
            hexa =>estad4,
            sseg =>db_estado
        ) ;

        HEX4: hexa7seg port map(
            hexa=>joga4,
            sseg=> db_jogada
        ) ;
        HEX5: hexa7seg port map(
            hexa=>lim4,
            sseg=> db_limite
        ) ;
end architecture;

```

## 2.3) Plano de testes

Cenário #1 – Acerto de todas as jogadas				
#	Operação	Sinais de Entrada	Resultado Esperado	Resultado Observado
c.i.	Condições Iniciais			
1	Iniciar ciclo	iniciar = 1		
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	igual = 1 db_limite = 8
10	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001, botoes = 0001, botoes = 0010	igual = 1 db_limite = 9	igual = 1 db_limite = 9
11	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001, botoes = 0001, botoes = 0010, botoes = 0010	igual = 1 db_limite = 10	igual = 1 db_limite = 10
12	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001, botoes = 0001, botoes = 0010, botoes = 0010, botoes = 0100	igual = 1 db_limite = 11	igual = 1 db_limite = 11

13	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001, botoes = 0001, botoes = 0010, botoes = 0010, botoes = 0100, botoes = 0100	igual = 1 db_limite = 12	igual = 1 db_limite = 12
14	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001, botoes = 0001, botoes = 0010, botoes = 0010, botoes = 0100, botoes = 0100, botoes = 1000	igual = 1 db_limite = 13	igual = 1 db_limite = 13
15	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001, botoes = 0001, botoes = 0010, botoes = 0010, botoes = 0100, botoes = 0100, botoes = 1000, botoes = 1000	igual = 1 db_limite = 14	igual = 1 db_limite = 14
16	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001, botoes = 0001, botoes = 0010, botoes = 0010, botoes = 0100, botoes = 0100, botoes = 1000, botoes = 1000, botoes = 0001	igual = 1 db_limite = 15	igual = 1 db_limite = 15
17	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001, botoes = 0001, botoes = 0010, botoes = 0010, botoes = 0100, botoes = 0100, botoes = 1000, botoes = 1000, botoes = 0001, botoes = 0100	igual = 1, fim = 1, acertou = 1	igual = 1, fim = 1, acertou = 1

Cenário #2 – Erro na 4ª jogada				
#	Operação	Sinais de Entrada	Resultado Esperado	Resultado Observado
c.i.	Condições Iniciais			
1	Iniciar Ciclo	iniciar = 1		
2	Acertar jogadas	botoes = 0001	igual = 1 db_limite = 1	igual = 1 db_limite = 1

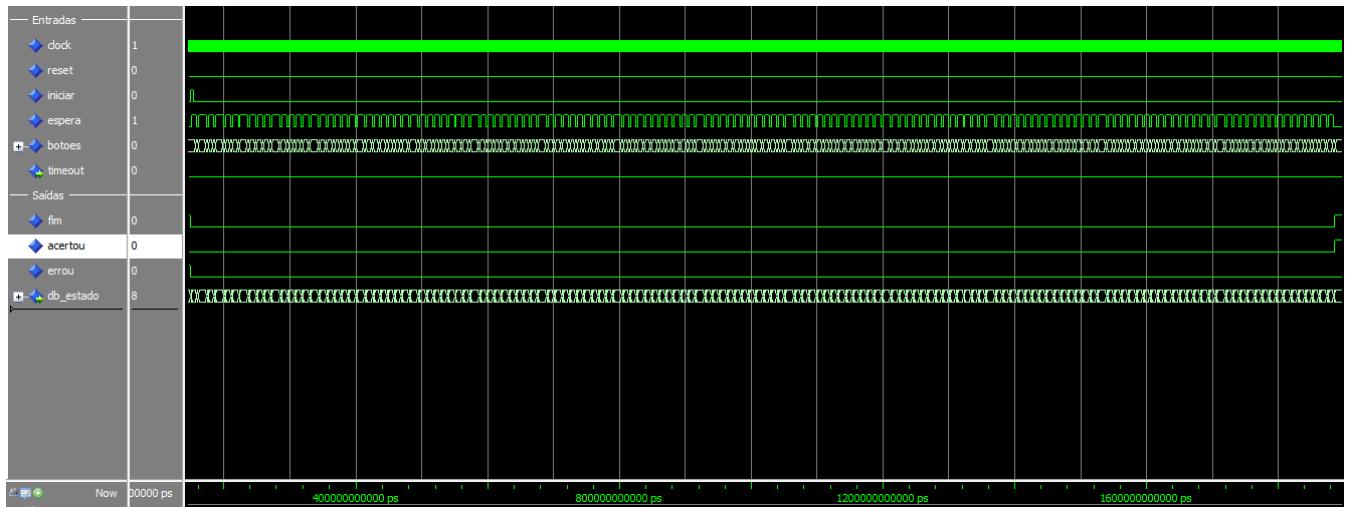
<b>3</b>		<b>botoes = 0001, botoes = 0010</b>	<b>igual = 1 db_limite = 2</b>	<b>igual = 1 db_limite = 2</b>
<b>4</b>		<b>botoes = 0001, botoes = 0010, botoes = 0100</b>	<b>igual = 1 db_limite = 3</b>	<b>igual = 1 db_limite = 3</b>
<b>5</b>	<b>Errar jogada</b>	<b>botoes = 0010</b>	<b>fim = 1, errou =1</b>	<b>fim = 1, errou =1</b>

#### Cenário #3 – Perder por timeout após 3ª jogada

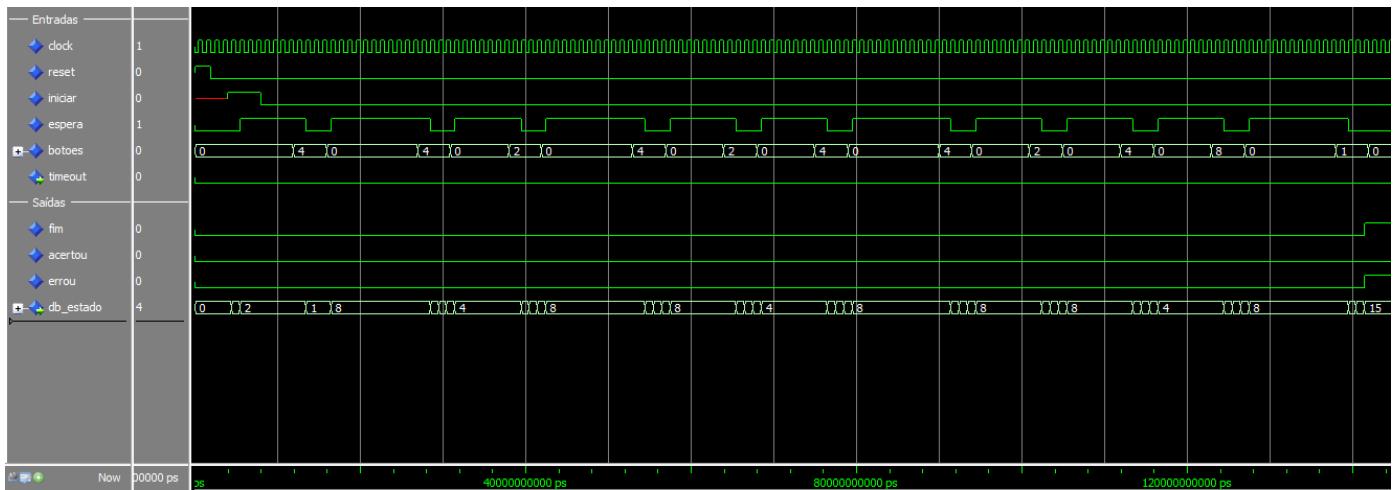
#	Operação	Sinais de Entrada	Resultado Esperado	Resultado Observado
c.i.	<b>Condições Iniciais</b>			
<b>1</b>	<b>Iniciar Ciclo</b>	<b>iniciar = 1</b>		
<b>2</b>	<b>Acertar jogadas</b>	<b>botoes = 0001</b>	<b>igual = 1 db_limite = 1</b>	<b>igual = 1 db_limite = 1</b>
<b>3</b>		<b>botoes = 0001, botoes = 0010</b>	<b>igual = 1 db_limite = 2</b>	<b>igual = 1 db_limite = 2</b>
<b>4</b>		<b>botoes = 0001, botoes = 0010, botoes = 0100</b>	<b>igual = 1 db_limite = 3</b>	<b>igual = 1 db_limite = 3</b>
<b>5</b>	<b>Esperar 6 segundos</b>		<b>fim = 1, errou =1</b>	<b>fim = 1, errou =1</b>

## 2.4) Simulação do circuito

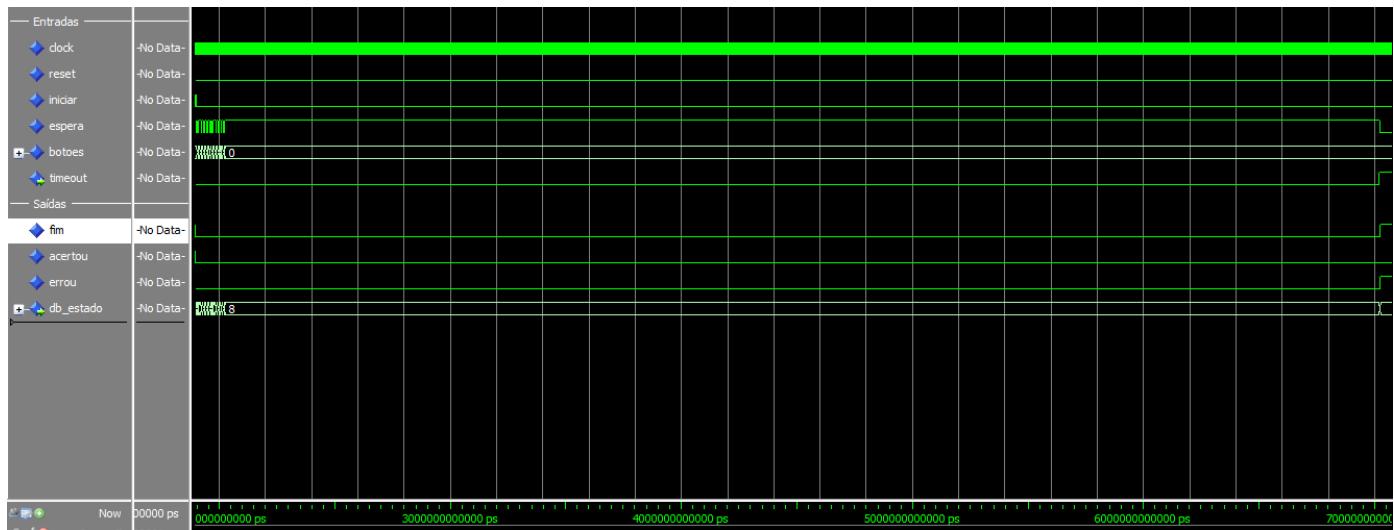
### 2.4.1) Cenário #1



#### 2.4.2) Cenário #2



#### 2.4.3) Cenário #3



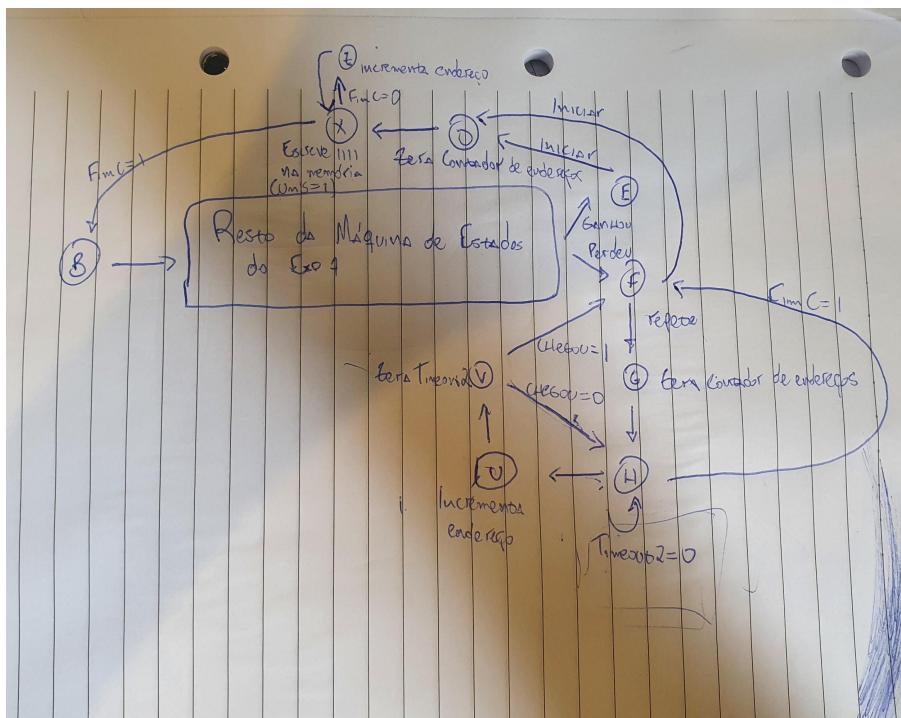
## 2.5) Designação de pinos

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
BTOES(0)	GPIO_0_D19	PIN_P17	StaticIO – Button 0/1 – DIO3
BTOES(1)	GPIO_0_D21	PIN_M18	StaticIO – Button 0/1 – DIO4
BTOES(2)	GPIO_0_D23	PIN_L17	StaticIO – Button 0/1 – DIO5
BTOES(3)	GPIO_0_D25	PIN_K17	StaticIO – Button 0/1 – DIO6

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_igual	Led LEDR4	PIN_N2	-
db_tem_jogada	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_estado	Display HEX5	[0] PIN_N9 [1] PIN_M8 [2] PIN_T14 [3] PIN_P14 [4] PIN_C1 [5] PIN_C2 [6] PIN_W19	-

### 3) Descrição do Desafio

O desafio consiste 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 G,H,U e V, que acionam a lógica de repetição, e os estados O,X e Z, 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 G, que zera o contador de endereços. Em seguida vai para o estado H, onde aguarda 2 segundos antes de ir pro estado U. No estado U ele incrementa o endereço e vai para o estado V, onde zera o contador de tempo do estado H. Se ele estiver no estado H 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 V 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 O, onde zera o contador de endereços e vai para o estado X, 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 Z, onde incrementa o endereço e volta para o estado X. Se ele estiver no X e chegar ao fim da memória ele volta para o início da máquina de estados. O diagrama a seguir ilustra a máquina de estados modificada:



Os códigos a seguir são referentes a Unidade de controle, fluxo de dados, circuito geral e contador de 2 segundos respectivamente:

```
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(3 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      (A,      B,      C,      D,      E,      F,
M,J,L,K,W,Q,P,R,S,T,I,G,H,U,V,O,Z,X);
        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
        R when Eatual=B else
        R when Eatual=R and jogada ='0' else
        S when Eatual=R and jogada ='1' else
        T when Eatual=S else
        I when Eatual=T else
        J when Eatual=I else
        J when Eatual=J and jogada='0' and timeout='0' else
        M when Eatual=J and jogada='1'and timeout='0' else
        C when Eatual=M else
        D when Eatual=C and enderecoIgualLimite='0' and igual='1' else
        F when Eatual=C and igual='0' else
        F when Eatual=J and timeout='1'else
        F when Eatual=Q and timeout='1' else
        L when Eatual=W else
        K when Eatual=C and enderecoIgualLimite='1' and igual='1' and
fimC='0' else
        Q when Eatual=K else
        Q when Eatual=Q and jogada ='0' and timeout='0' else
        P when Eatual=Q and jogada ='1' and timeout='0' else
        W when Eatual=P else
        J when Eatual=D else
        J when Eatual=L else

```

```

E when Eatual=C and FimC='1' and igual ='1' else
O when Eatual=E and iniciar='1' else
E when Eatual=E and iniciar='0' else
O when Eatual=F and iniciar='1' and repete='0' else
F when Eatual=F and iniciar='0' and repete='0'else
    X when Eatual =O else
    Z when Eatual=X and fimC='0' else
    B when Eatual =X and fimC='1' else
    X when Eatual=Z else
G when Eatual=F and repete='1' else
H when Eatual=G else
H when Eatual = H and timeout2 ='0' and fimC='0' else
    F when Eatual=H and timeout2 ='0' and fimC='1' else
U when Eatual =H and timeout2='1' else
V when Eatual =U else
H when Eatual=V and chegou='0' else
F when Eatual=V and chegou='1' else
A;

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

with Eatual select
  zera_T <= '1' when K | L | D | B | I,
    '0' when others;
with Eatual select
  zera_2 <= '1' when B | V,
    '0' when others;

with Eatual select
  reset_m <='1' when Z | X,
    '0' when others;

with Eatual select
  conta_T <= '1' when J | Q,
    '0' when others;
with Eatual select

```

```

conta_2 <= '1' when H,
          '0' when others;

with Eatual select
espera <= '1' when Q | R,
          '0' when others;

with Eatual select
escreve <= '0' when A | B | C | D | E | F | J,
          '1' when W | T | X,
          '0' when others;

with Eatual select

zera <=  '0' when A | C | D | E | F | M | J,
          '1' when B | L | I | G | O,
          '0' when others;

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

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

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

with Eatual select
pronto <= '0' when A | B | C | D | M | J,
          '1' when E | F | G | H | U | V,
          '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 | G | H | U | V,
                '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
                "0100" when Q,  --Q,4
                "0100" when U,  --Q,4
                "0010" when R,  --R,2
                "0010" when G,
                "0011" when K,  --K,3
                "0011" when H,  --K,3
                "0101" when P,  -- P,5
                "0110" when W,  --W,6
                "0111" when M,  -- M,7
                "1000" when J,  -- J,8
                "1001" when L,  --L,9
                "1111" when F,      -- F
                "0001" when others;  -- 1
end fsm;

```

```

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
);
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_EQQB   : in  std_logic;
-- saidas
o_AGTB   : out std_logic;
o_ALTB   : out std_logic;
o_EQQB   : 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_outc,or_J
GD,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);
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
);
fimE<=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=> "00000000000000",
    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=> "000000000000",
    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_EQB  => '1',
    -- saidas
    o_ALTB =>enderecoMenorQueLimite,

```

```

    o_EQB => IgualLimite
);

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_EQB => '1',
    -- saidas
    o_AGTB =>great,
    o_ALTB =>menor,
    o_EQB => 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_EQB => '1',
    -- saidas
    o_EQB => fimRes
);

```

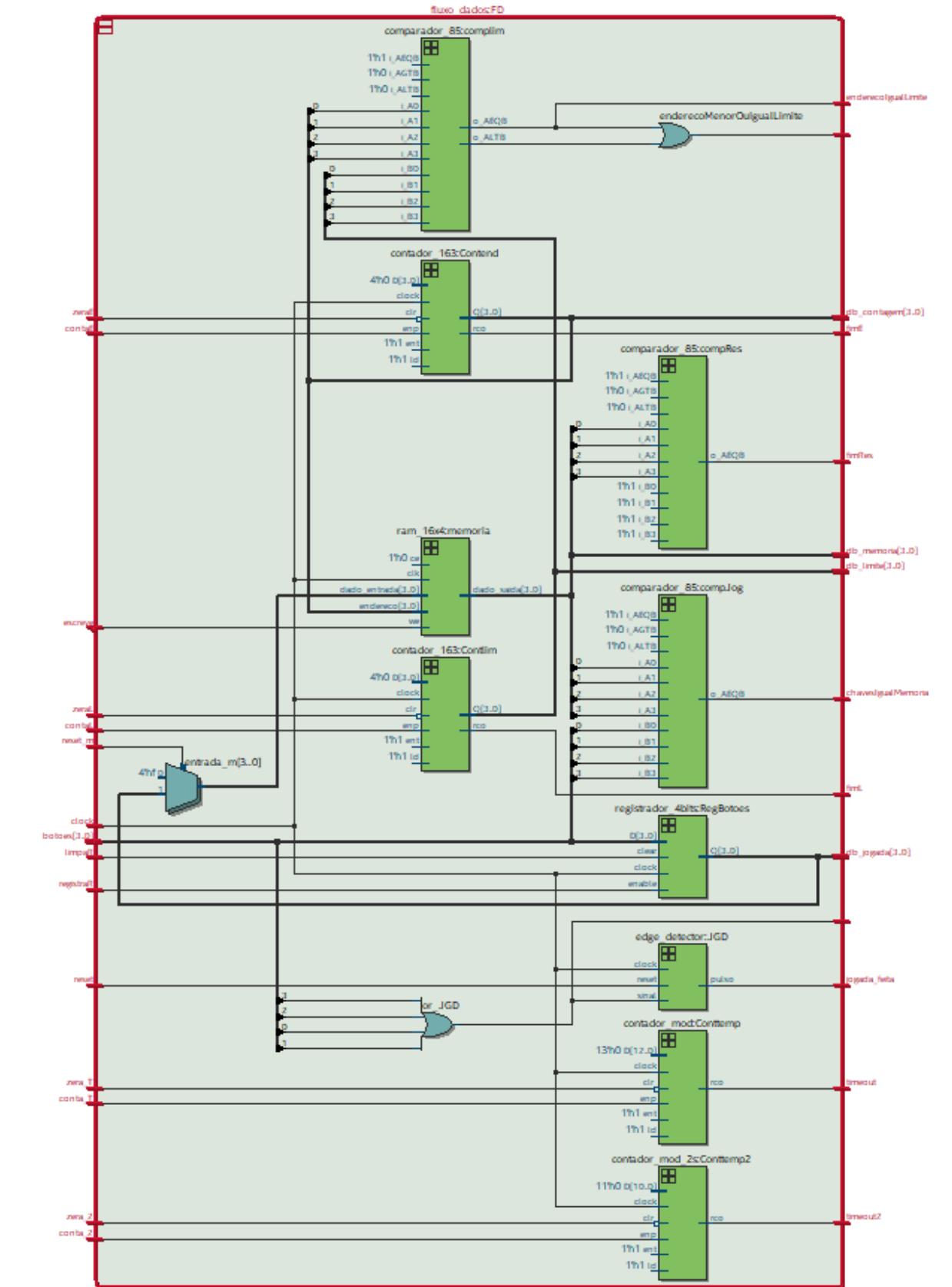
```

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';
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;

```

Segue também o diagrama do fluxo de dados modificado com o multiplexador na entrada da memória:



```

library ieee;
use ieee.std_logic_1164.all;
entity circuito_exp7_desafio 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;
        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_exp7_desafio 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
);
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(3 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;

signal conta4,memo4,estad4,joga4,lim4,botoes_led: std_logic_vector (3
downto 0);
signal
conta,zeraE,registra,clk,jogada,igual_i,escreve_baixo,escreve,db_tem_jogad
a,db_clock: std_logic;
signal
zeraL,contaL,contaE,fimE,fimL,jogada_feita,chavesIgualMemoria,enderecoIgu
alLimite_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);
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) xor mostra_memoria(0);
    leds(1)<=botoes(1) xor mostra_memoria(1);
    leds(2)<=botoes(2) xor mostra_memoria(2);
    leds(3)<=botoes(3) 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
);
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: hexa7seg port map(
    hexa =>estad4,
    sseg =>db_estado
);

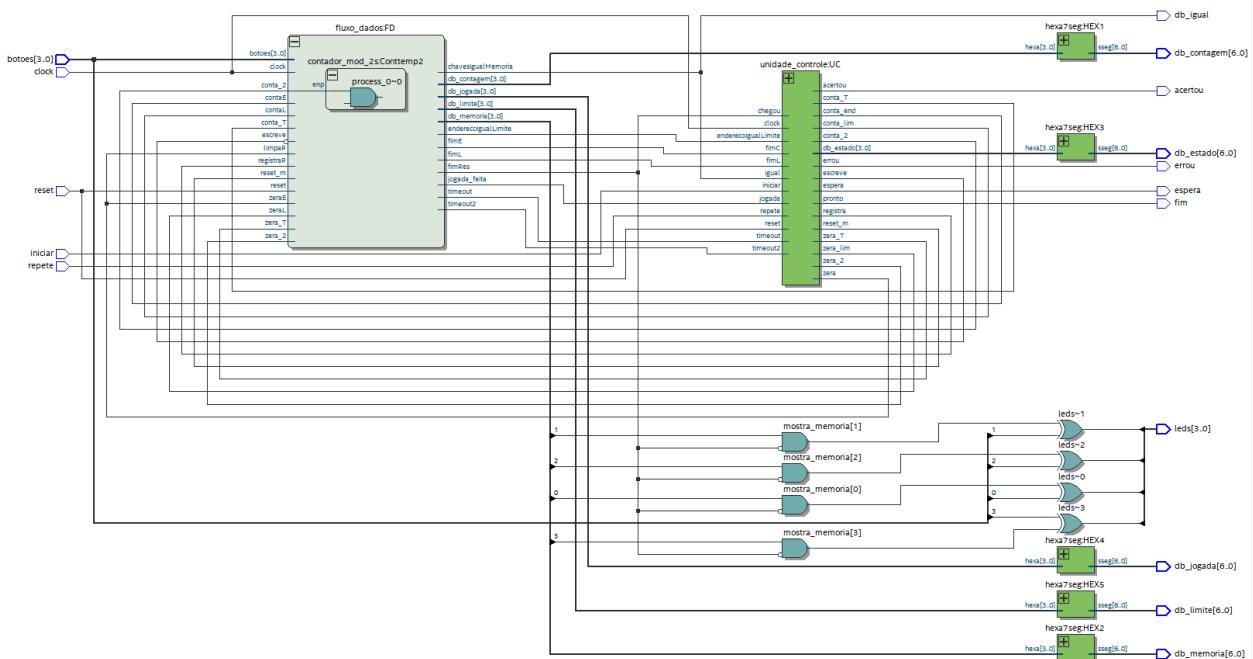
```

```

HEX4: hexa7seg port map (
    hexa=>joga4,
    sseg=> db_jogada
);
HEX5: hexa7seg port map (
    hexa=>lim4,
    sseg=> db_limite
);
end architecture;

```

Novo diagrama do circuito geral:



```

-- 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

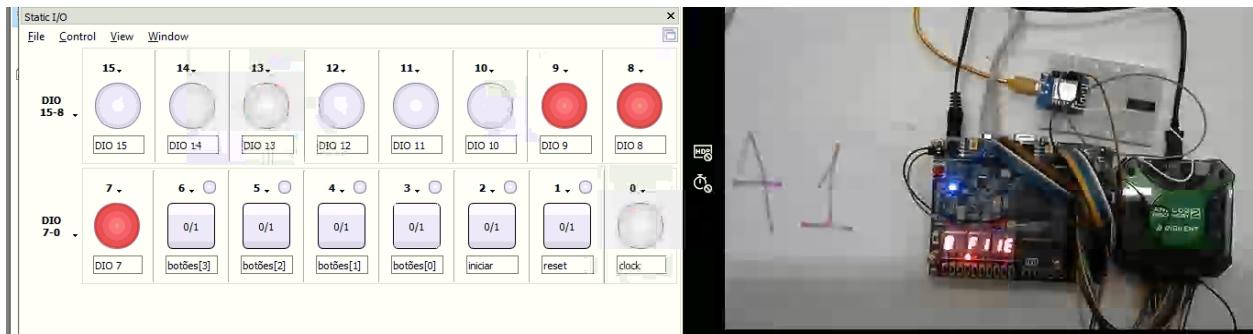
```

## 4) Demonstração do Funcionamento dos Circuitos

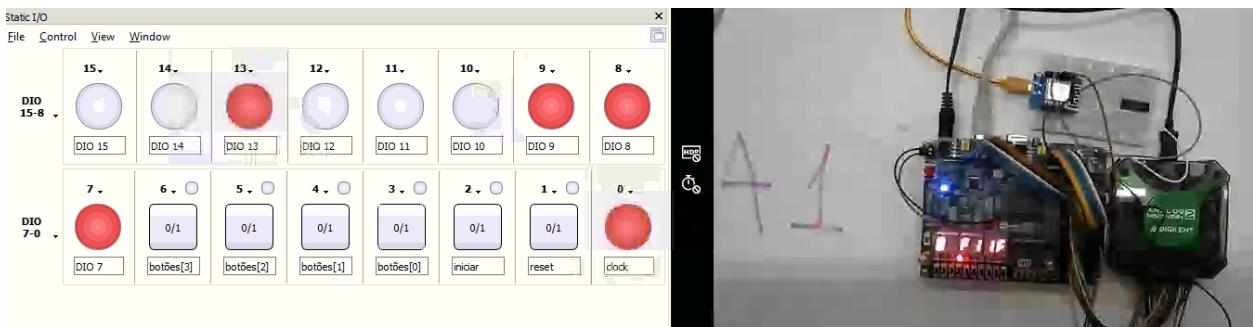
### 4.1) Circuito “circuito\_exp7.vhd”

#### 4.1.1) Acerto de todas as jogadas

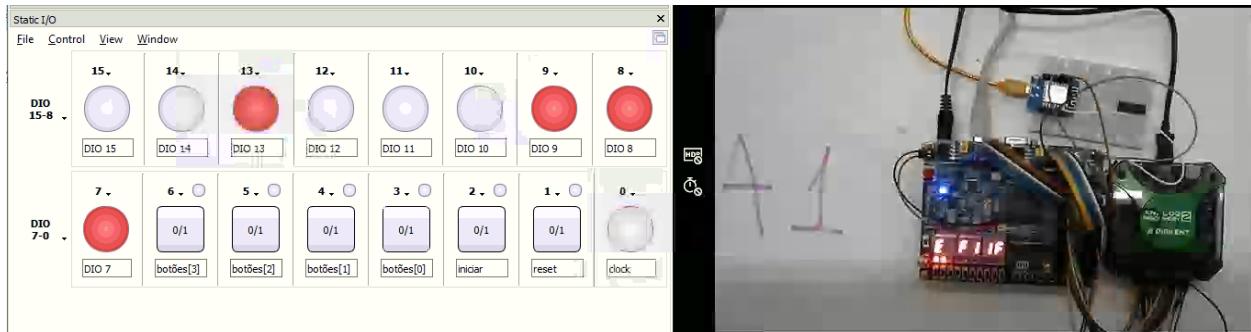
Neste caso, a fim de simplificar a demonstração do funcionamento do circuito, anexaremos as imagens relativas às últimas jogadas da última rodada do Jogo do Desafio da Memória.



Após 15 rodadas corretas e 14 jogadas da última rodada corretas, o circuito encontra-se na posição de memória E.

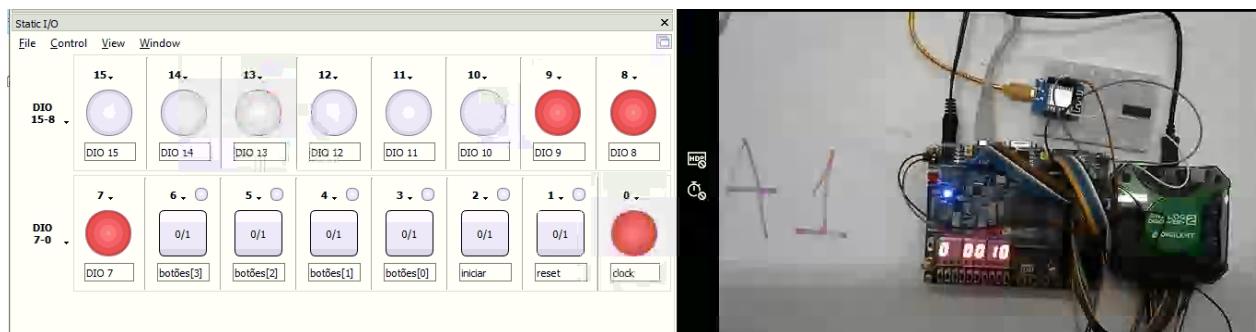


Após uma jogada correta, o circuito encontra-se na última posição de memória.

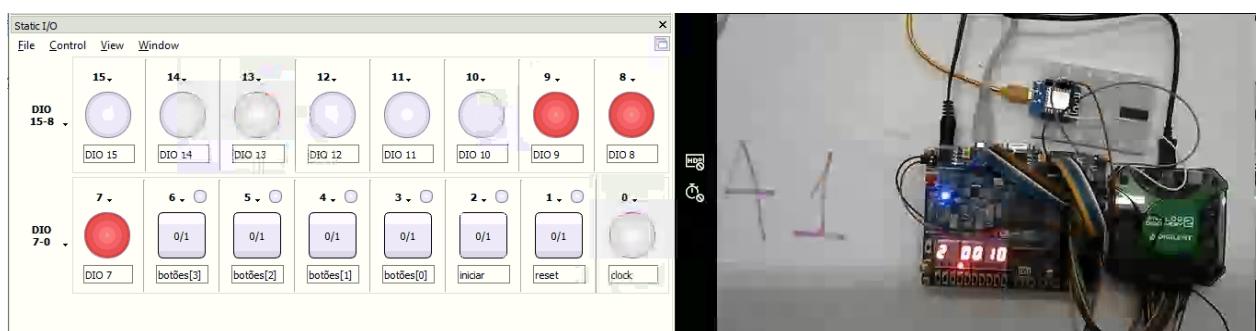


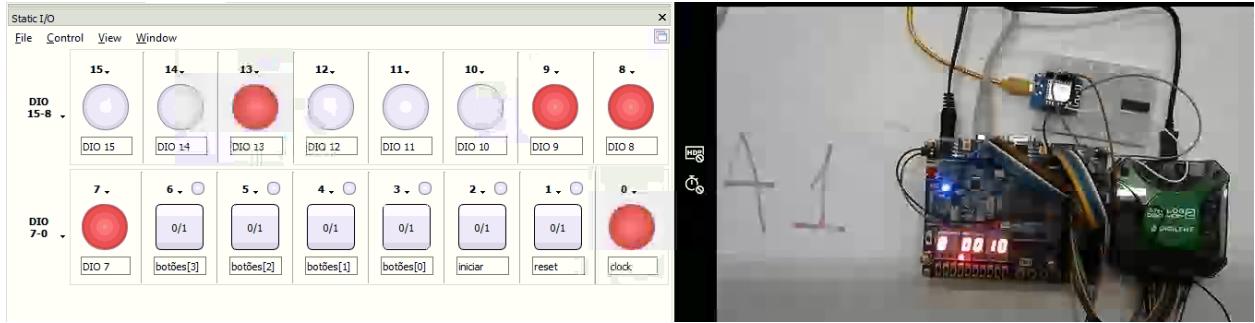
Após acertar a última jogada, o circuito vai para o estado de acerto (E) e acende os LEDs ligados aos sinais “fim” e “acertou”

#### 4.1.2) Erro na 4<sup>a</sup> rodada

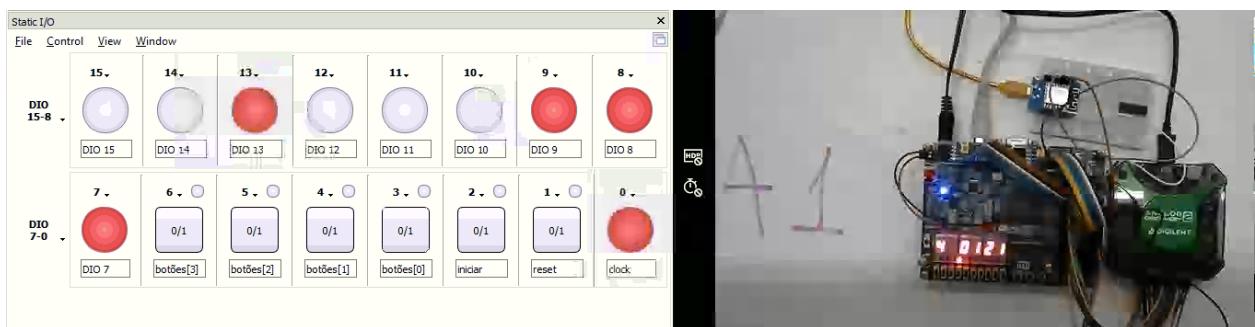


Condições iniciais

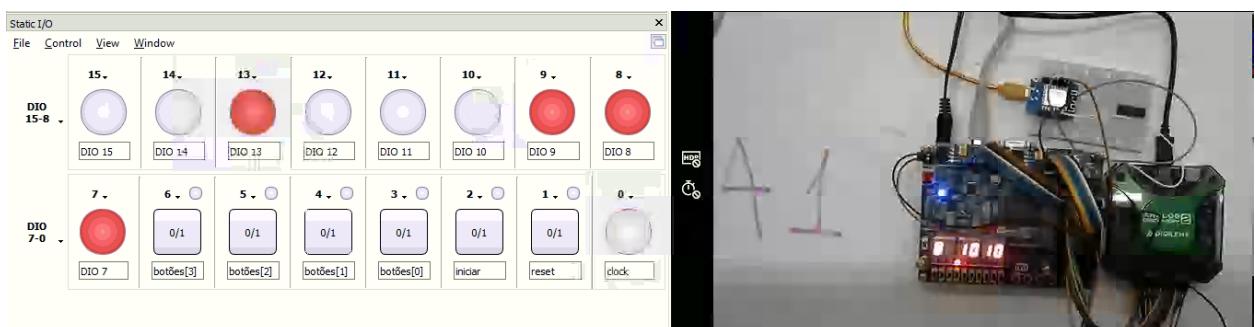




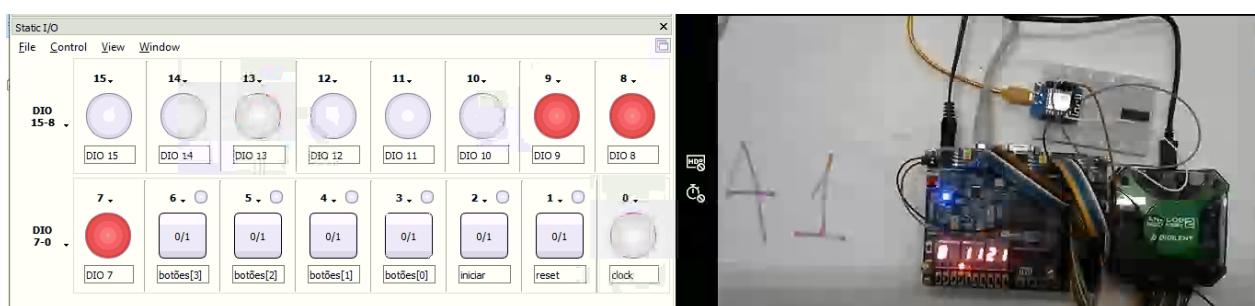
Rodada 1 - Escrita da primeira jogada



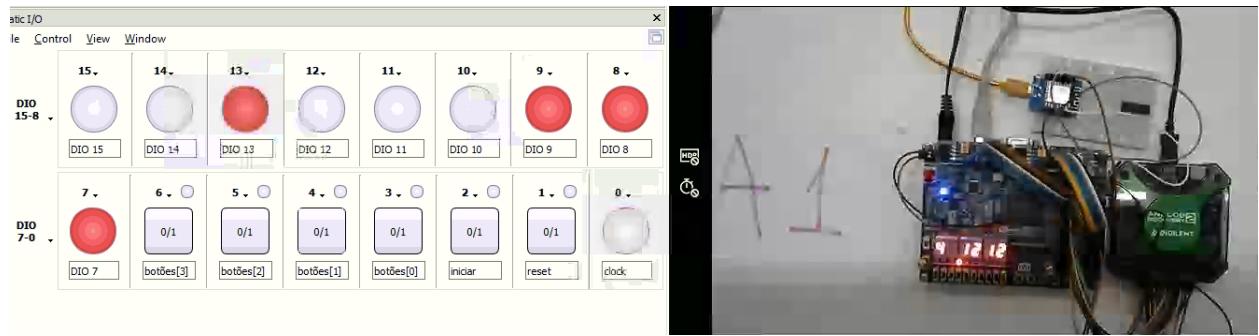
Rodada 2 - 1<sup>a</sup> jogada correta



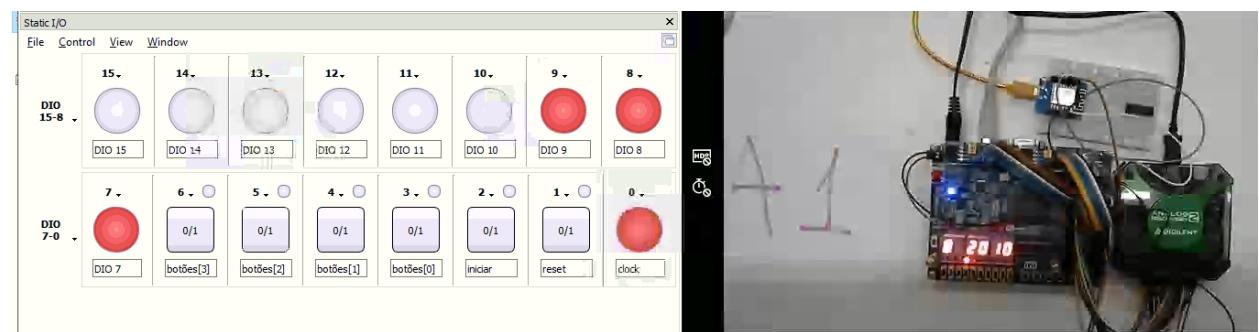
Rodada 2 - Escrita da 2<sup>a</sup> jogada



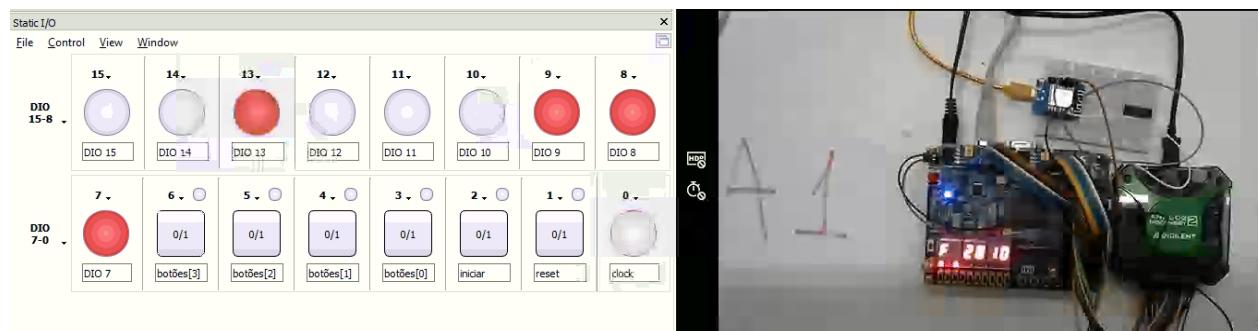
Rodada 3 - 1<sup>a</sup> jogada correta



Rodada 3 - 2<sup>a</sup> jogada correta

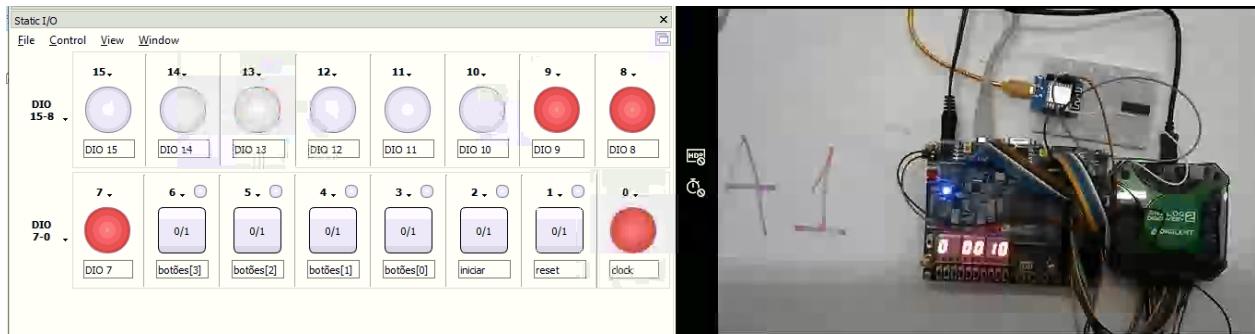


Rodada 3 - Escrita da 3<sup>a</sup> jogada

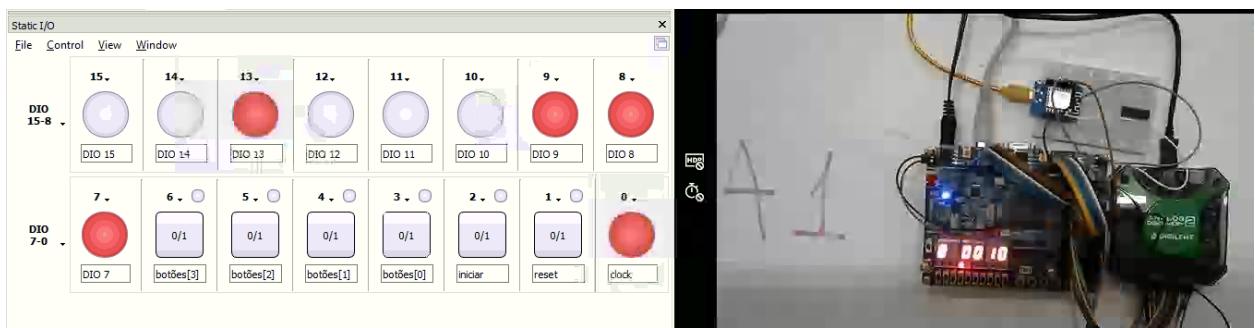
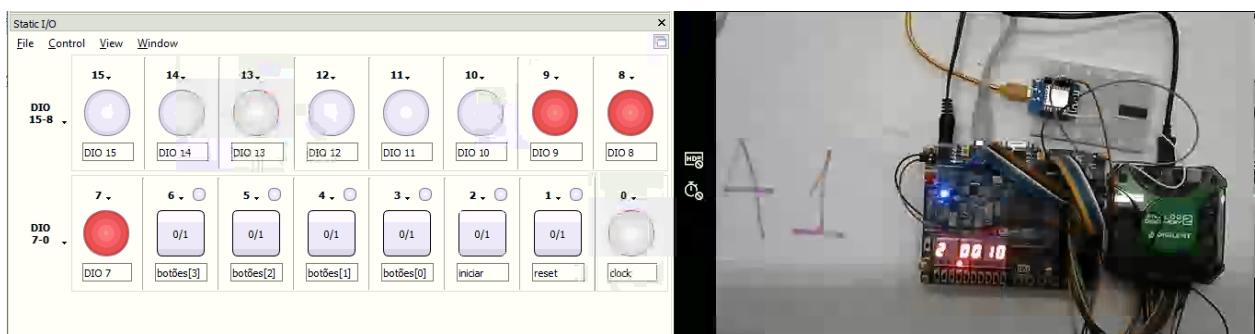


Rodada 4 - Erro na 1<sup>a</sup> jogada. O circuito vai para o estado de erro (F) e acende os LEDs ligados aos sinais “” e “errou”

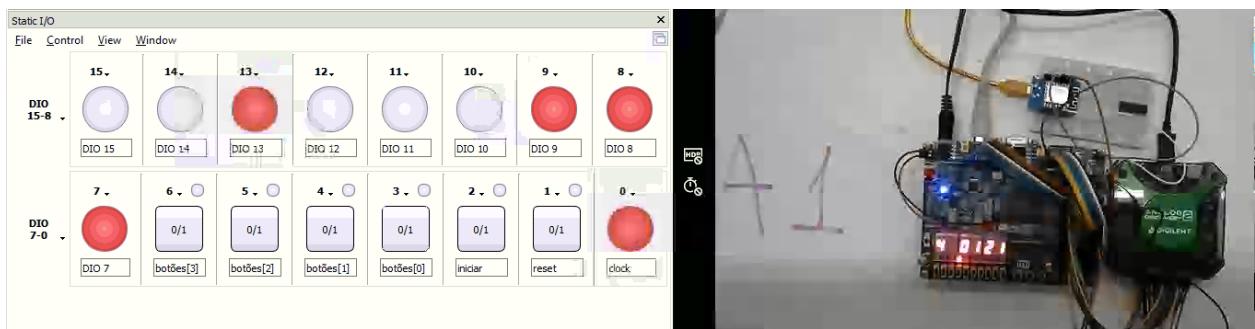
#### 4.1.3) Timeout na 4<sup>a</sup> rodada



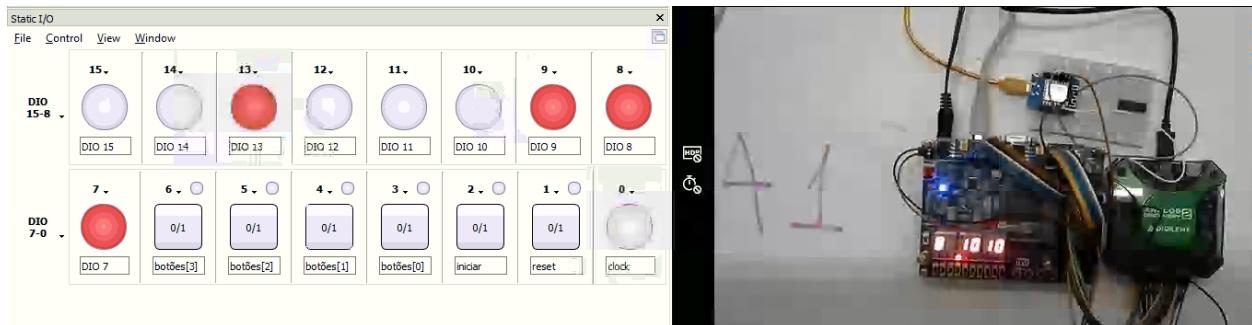
Condições iniciais



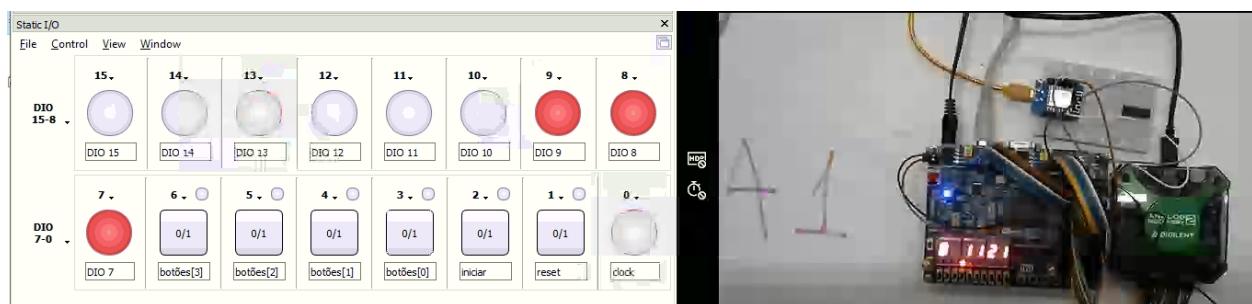
Rodada 1 - Escrita da primeira jogada



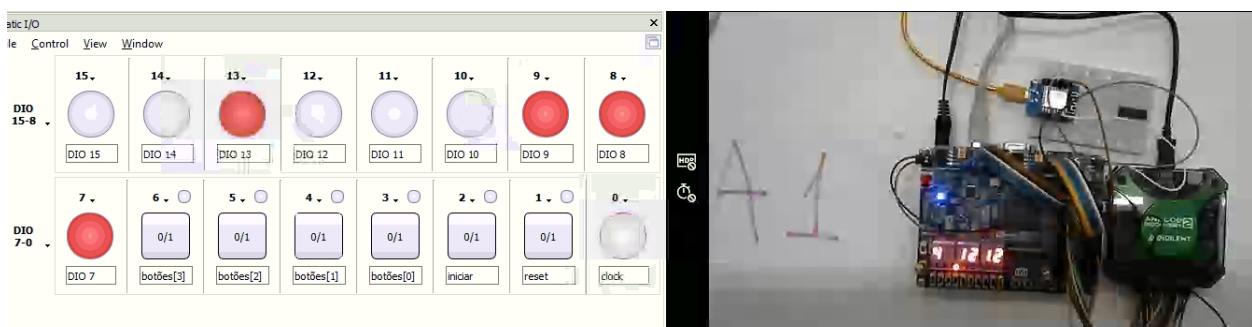
## Rodada 2 - 1<sup>a</sup> jogada correta



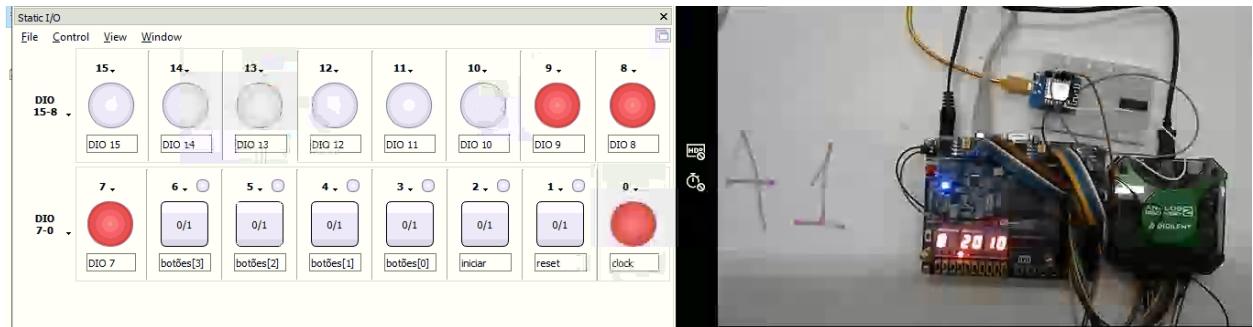
## Rodada 2 - Escrita da 2<sup>a</sup> jogada



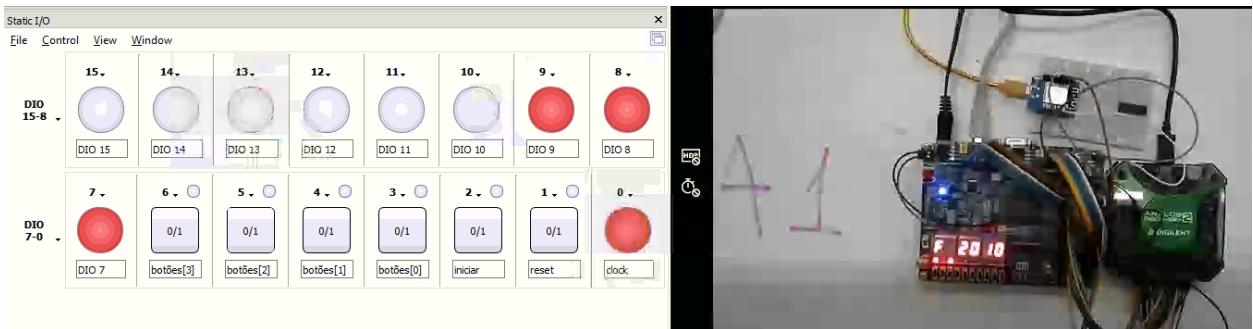
## Rodada 3 - 1<sup>a</sup> jogada correta



## Rodada 3 - 2<sup>a</sup> jogada correta



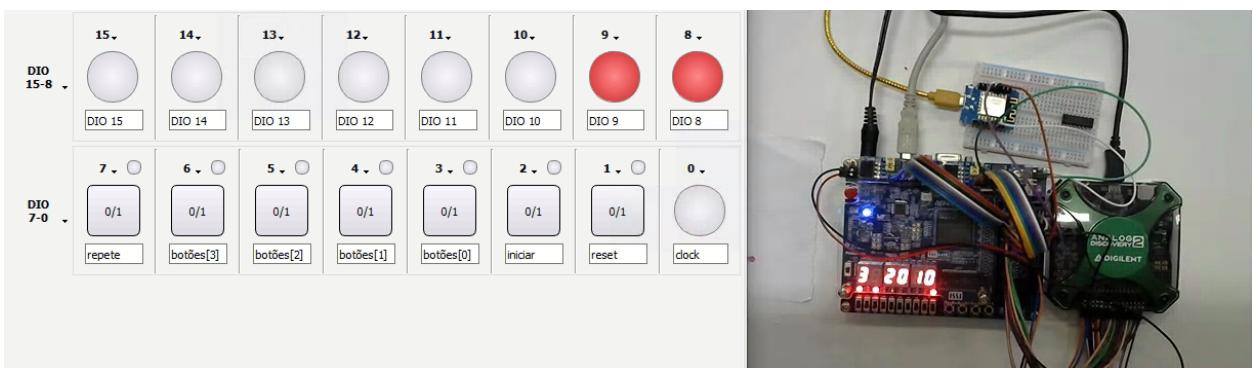
Rodada 3 - Escrita da 3<sup>a</sup> jogada



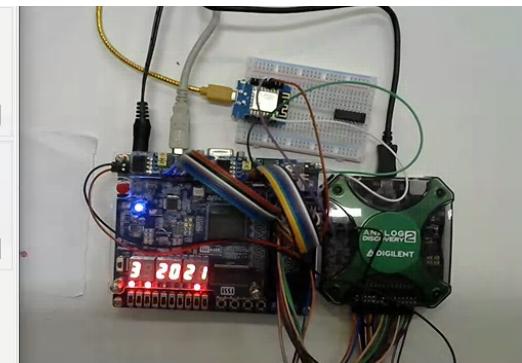
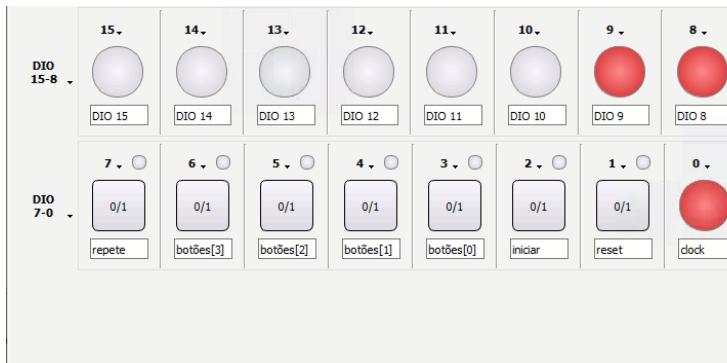
Rodada 4 - Após aguardar 5s, o circuito dá *timeout*. O circuito então vai para o estado de erro (F) e acende os LEDs ligados aos sinais “” e “errou”

#### 4.2) Circuito “circuito\_exp7\_desafio.vhd”

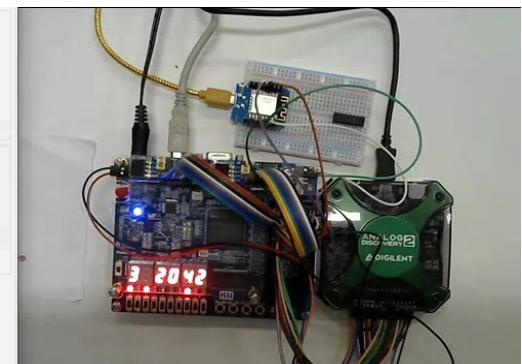
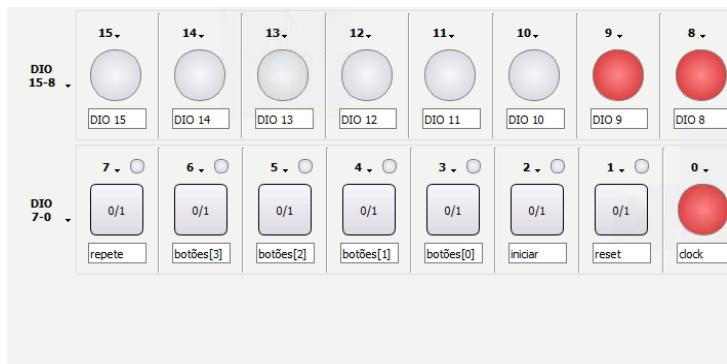
A execução das 3 primeiras rodadas foi igual a dos item 3.1.2 e 3.1.3. Abaixo está a descrição do funcionamento do botão “repete”.



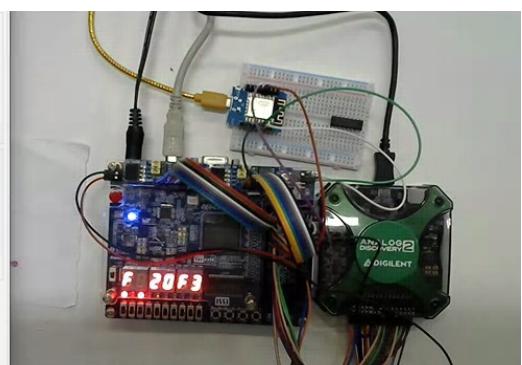
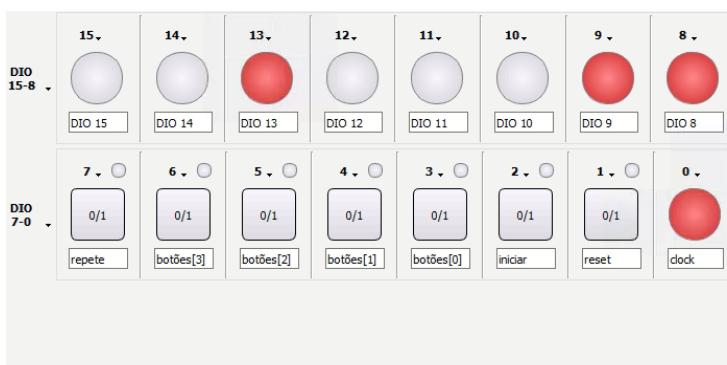
## Exibição da primeira jogada nos LEDs



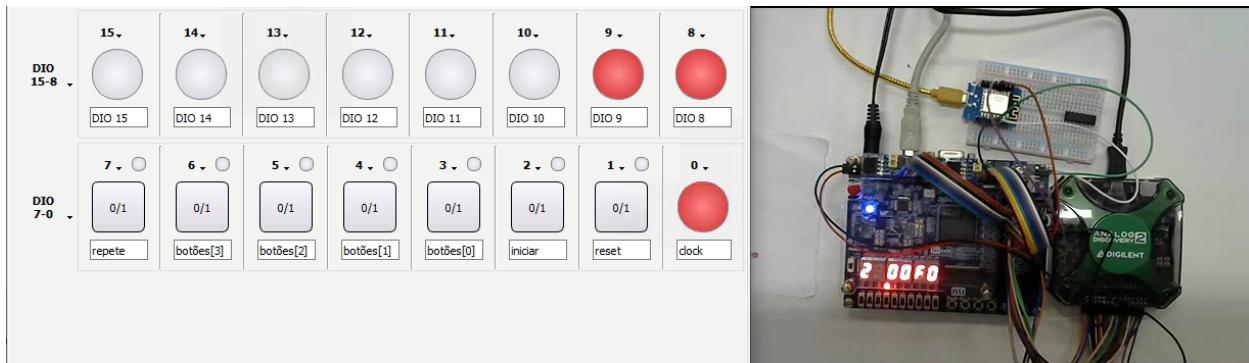
## Exibição da segunda jogada nos LEDs



## Exibição da terceira jogada nos LEDs



Após exibir todas as jogadas, o circuito volta para o estado F, onde o botão “repete” pode ser acionado novamente



Após apertar o botão “iniciar”, o circuito reseta a memória e volta para o estado R

## 5) Resultados Alcançados

### 5.1) Pontos Positivos

De todas as experiências que realizamos até o momento, esta foi a que mais nos desafiou e exigiu uma análise mais delicada para encontrarmos uma solução que cumprisse todos os requisitos esperados. Apesar das dificuldades encontradas, principalmente na resolução do desafio, a experiência correu bem e conseguimos cumprir todos os objetivos pretendidos, absorvendo devidamente os conteúdos propostos.

### 5.2) Pontos Negativos

Nesta experiência, tivemos alguns problemas durante a resolução do desafio. Na lógica que utilizamos, era necessário que houvesse o valor FFFF (valor inicial) na posição após a última que havíamos escrito. Para isso, precisávamos resetar a memória toda vez que o sinal “iniciar” fosse ativado. Tentamos fazer esse reset de diversas formas, obtendo sucesso ao fazê-lo através da adição de mais 3 estados na unidade controle.

### 5.3) Lições Aprendidas

No desafio desta experiência, acabamos tentando implementar uma solução um pouco mais rebuscada em pouco tempo. Por isso, acabamos não nos atentando a detalhes, o que viria a nos causar problemas na resolução do exercício.

## 6) Referências Técnicas

1. Apostila e apresentação da Experiência 7, disponibilizados no ambiente da disciplina PCS3635 no site E-disciplinas.
2. Tabela de Pinos da placa DE0-CV