



Laboratório Digital I - PCS3635

Relatório da Experiência 6:

Projeto Base 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

26/02/2021

1) Objetivo

Os objetivos da aula consistem em aprender sobre:

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

2) Elaboração do Circuito “circuito_exp6.vhd”

2.1) Descrição do funcionamento do circuito

Este circuito conta com boa parte das funcionalidades do Jogo do Desafio da Memória, apresentando um ciclo completo de jogadas, onde a cada rodada é acrescentada 1 jogada a mais.

Para exemplificar o funcionamento do circuito, utilizaremos as 3 primeiras rodadas do jogo. Na memória utilizada, possuímos nos 3 primeiros endereços os valores 0001, 0010 e 0100. As 3 primeiras rodadas seriam, portanto:

- 1 - 0001
- 2 - 0001, 0010
- 3 - 0001, 0010, 0100

Este ciclo se mantém até que a memória inteira seja percorrida ou o jogador erre uma jogada.

O circuito é composto de uma unidade de controle e um fluxo de dados com 2 contadores, 2 comparadores, 1 memória, 1 registrador, conversores para 7 segmentos e 1 detector de jogadas. O circuito basicamente segue uma lógica de ter um contador limitando a contagem de outro contador que incrementa os endereços da memória. Isso é feito usando um comparador que verifica se o endereço é menor que o limite. Quando o endereço é igual ao limite, ele incrementa 1 ao contador de limite e repete a lógica do circuito.

2.2) Descrição VHDL do circuito “circuito_exp6.vhd”

O código da unidade de controle segue a seguinte lógica:

```
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;
        jogada:   in std_logic;
        enderecoIgualLimite: in std_logic;
        zera:      out std_logic;
        conta_end:    out std_logic;
        conta_lim:   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
    );
end entity;

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

        -- logica de proximo estado
        Eprox <=
            A when Eatual=A and iniciar='0' else
            B when Eatual=A and iniciar='1' else
            J when Eatual=B else
            J when Eatual=J and jogada='0' else
            M when Eatual=J and jogada='1' else
            C when Eatual=M else
            D when Eatual=C and enderecoIgualLimite='0' and igual='1' else
            L;
    end;
end;

```

```

F when Eatual=C and igual='0' else
      L when Eatual=C and enderecoIgualLimite='1' and igual ='1'
and fimC='0'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,
                  '0' when others;
with Eatual select
    zera <= '0' when A | C | D | E | F | M | J,
                  '1' when B | L,
                  '0' when others;
with Eatual select
    conta_end <= '0' when A | B | C | E | F | M | J,
                  '1' when D,
                  '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;

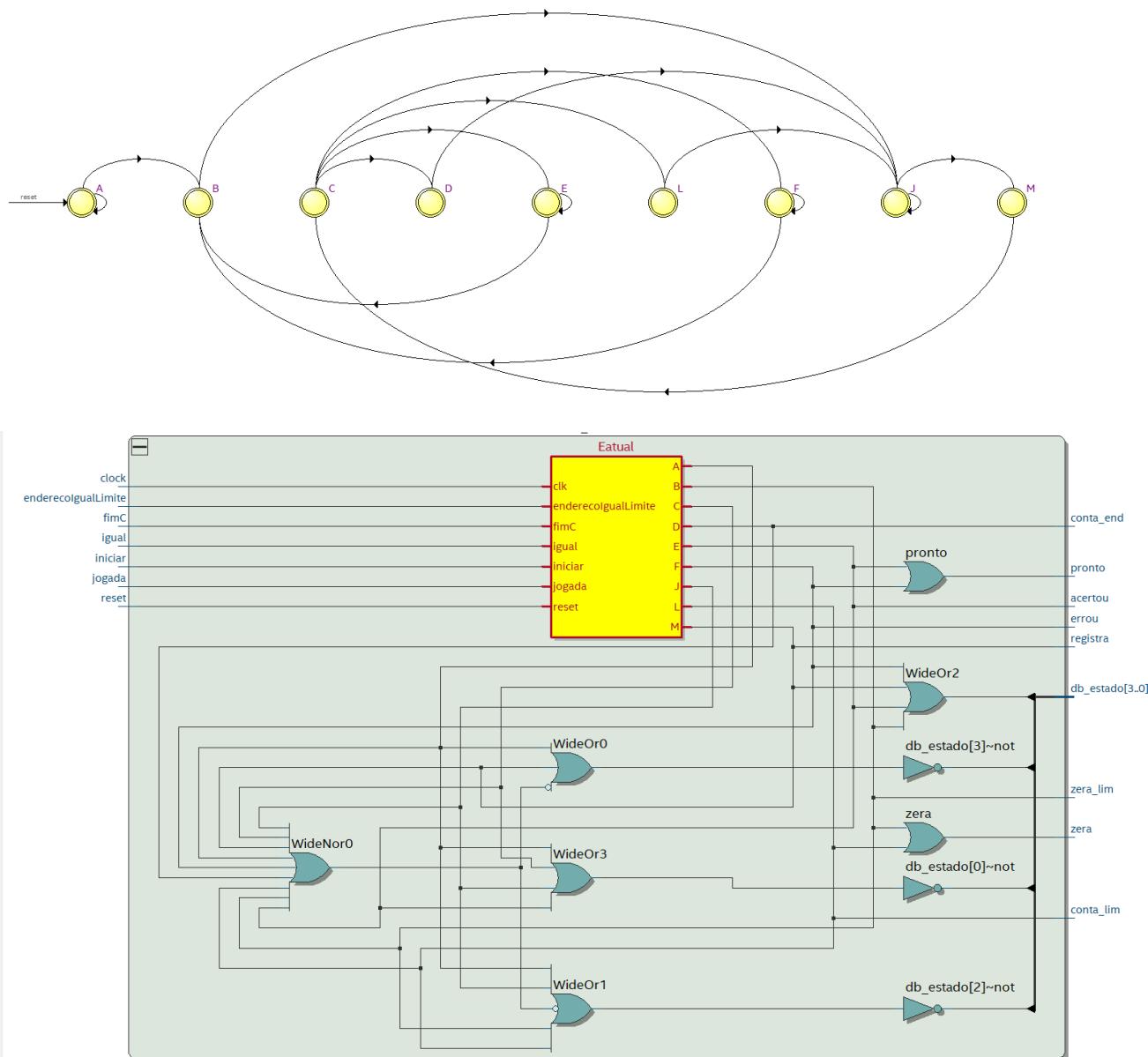
with Eatual select
    acertou <='0' when A | B | C | D | F | M | J,
        '1' when E,
        '0' when others;

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
                "0111" when M,          -- M
                "1000" when J,          -- J
                "1001" when L,
                "1111" when F,          -- F
                "0001" when others;     -- 1
end fsm;

```

Diagrama:



O fluxo de dados segue a seguinte lógica:

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

entity fluxo_dados is
    port (
        clock : in std_logic;
        reset : in 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 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_AGBTB : in  std_logic;
      i_ALTB  : in  std_logic;
      i_EQQB  : in  std_logic;
      -- saídas
      o_AGBTB : 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_out
c,or_JGD,zeraL_baixo,rco_outL,fim_L: std_logic;
signal enderecoMenorqueLimite, IgualLimite: 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;

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_AGBTB =>'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_AGBTB =>'0',

```

```

        i_ALTB => '0',
        i_AEQB => '1',
        -- saidas
        o_AGTB =>great,
        o_ALTB =>menor,
        o_AEQB => igual_out
    );

```



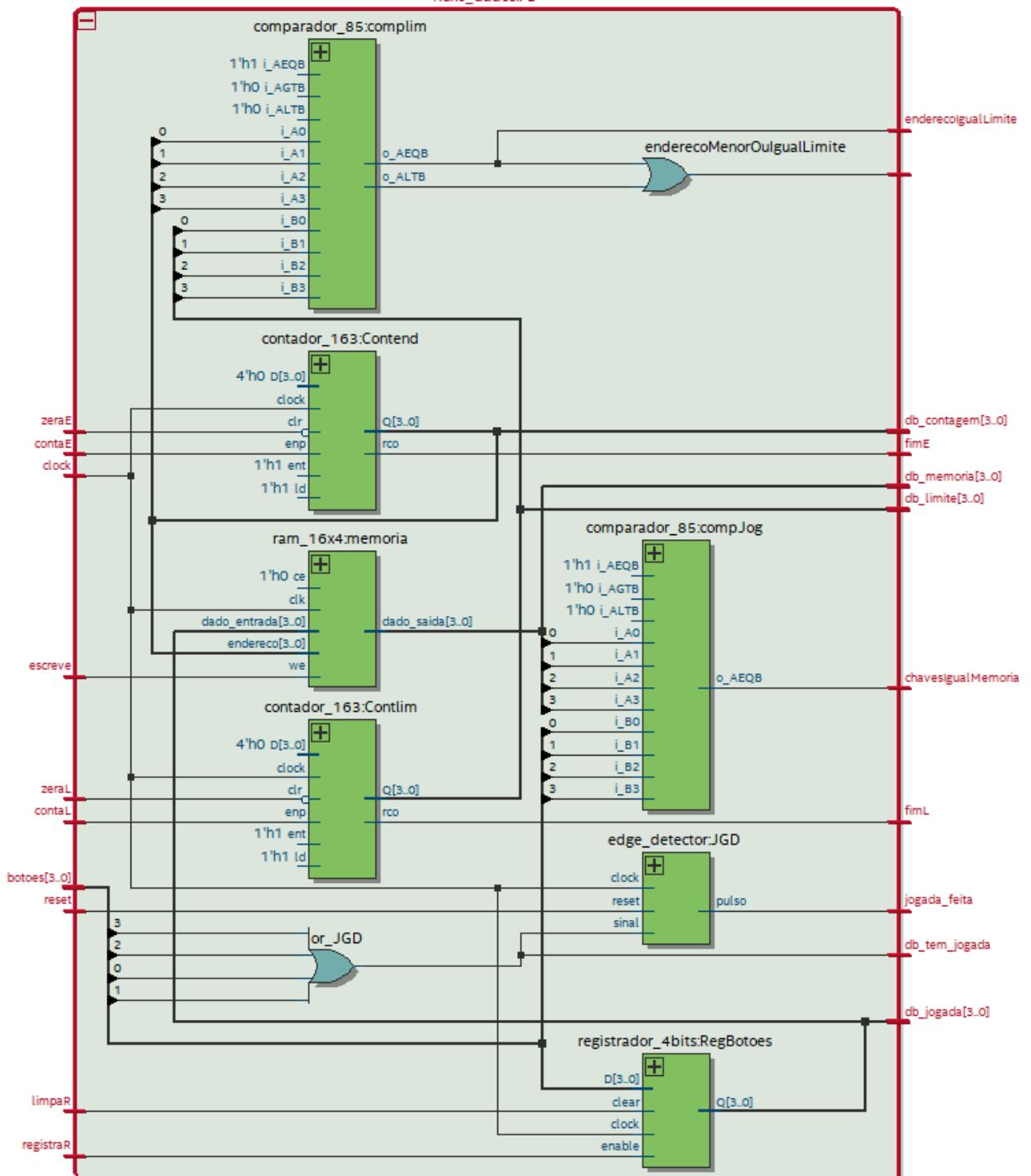
```

memoria: ram_16x4 port map(
    clk=> clock,
    endereco => s_endereco,
    dado_entrada => 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;

```

Diagrama:



E por fim o circuito segue:

```
library ieee;
use ieee.std_logic_1164.all;
entity circuito_exp6 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;
        pronto : 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);
        db_clock : out std_logic;
        db_tem_jogada : out std_logic
    );
end entity;

architecture estrutural of circuito_exp6 is
    component fluxo_dados is
        port(
            clock : in std_logic;
            reset: in 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;
        );
    end component;
    signal s_zeraE, s_limpaR, s_registraR, s_zeraL, s_contaL, s_contaE, s_escreve: std_logic;
    signal s_fimE, s_fimL: std_logic;
    signal s_db_limite, s_db_igual, s_db_contagem, s_db_memoria, s_db_estado, s_db_jogada, s_db_clock, s_db_tem_jogada: std_logic_vector(6 downto 0);
    signal s_leds: std_logic_vector(3 downto 0);
    signal s_acertou, s_errou, s_pronto: std_logic;
begin
    U1: fluxo_dados
        port map(
            clock => clock,
            reset => reset,
            zeraE => zeraE,
            limpaR => limpaR,
            registraR => registraR,
            zeraL => zeraL,
            contaL => contaL,
            contaE => contaE,
            escreve => escreve,
            botoes => botoes,
            fimE => s_fimE,
            fimL => s_fimL
        );
    end;
```

```

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;
jogada:     in std_logic;
enderecoIgualLimite: in std_logic;
zera:        out std_logic;
conta_end:   out std_logic;
conta_lim:   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
);
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,fim,zeraE,registra,clk,jogada,igual_i: std_logic;
                                signal
zeraL,contaL,contaE,fimE,fimL,jogada_feita,chavesIgualMemoria,endere
coIgualLimite: 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=>'1',
        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
    );
    db_igual<=chavesIgualMemoria;
    UC: unidade_controle port map(
        clock=>clock,
        reset=>reset,
        iniciar=>iniciar,

```

```

fimC=>fimE,
fimL=>fimL,
jogada=>jogada_feita,
enderecoIgualLimite=>enderecoIgualLimite,
zera=>zeraE,
conta_end=>contaE,
conta_lim=>contaL,
zera_lim=>zeraL,
pronto=>pronto,
db_estado=>estad4,
acertou=>acertou,
errou=>errou,
registra=>registra,
igual=>chavesIgualMemoria
);
db_clock<=clk;
HEX1: hexa7seg port map(
    hexa =>conta4,
    sseg =>db_contagem
);

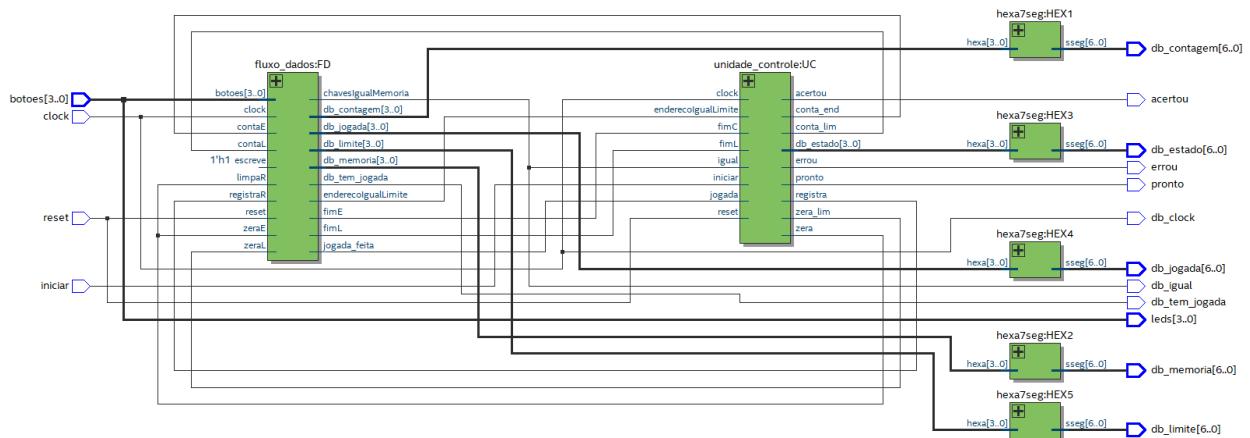
HEX2: hexa7seg port map(
    hexa =>memo4,
    sseg =>db_memoria
);

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;

```

Diagrama Geral:



2.3) Plano de testes

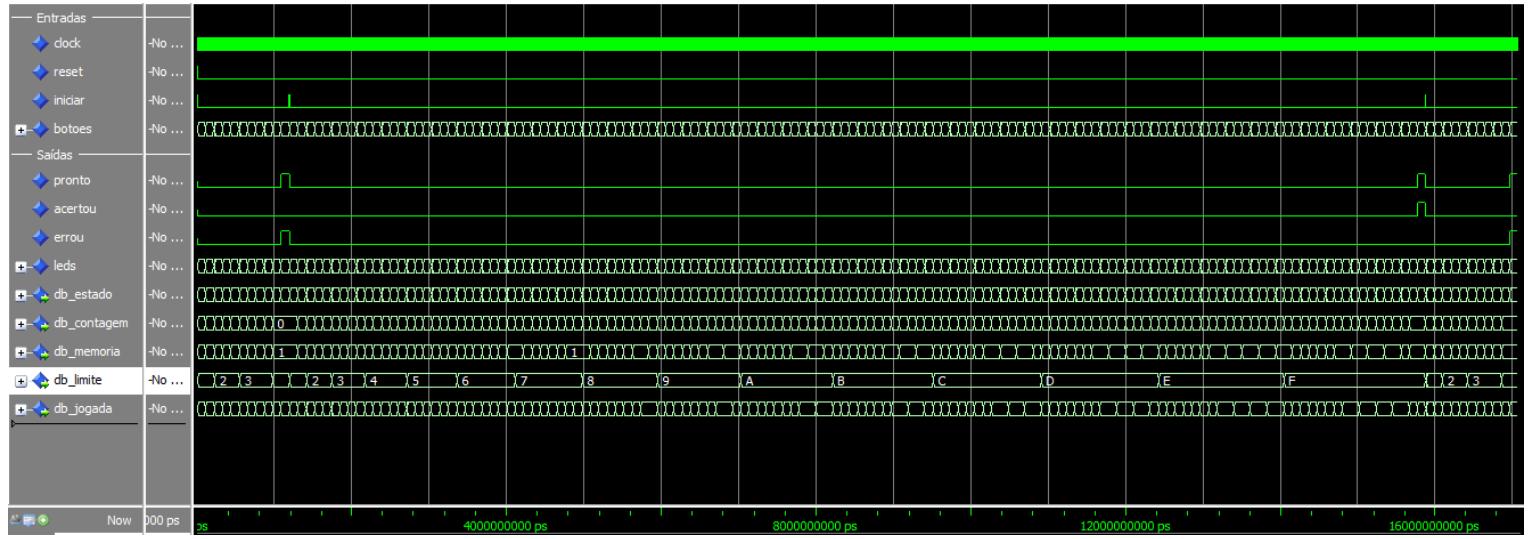
Cenário #1 – Acerto de todas as jogadas				
#	Operação	Sinais de Entrada	Resultado Esperado	Resultado Observado
c.i.	Condições Iniciais			
1	Iniciar ciclo	iniciar = 1		
2	Acertar jogada	botoes = 0001	igual = 1	
3	Acertar jogada	botoes = 0001, botoes = 0010	igual = 1	
4	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100	igual = 1	
5	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000	igual = 1	
6	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100	igual = 1	

7	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010	igual = 1	
8	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001	igual = 1	
9	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001, botoes = 0001	igual = 1	
10	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001, botoes = 0001, botoes = 0010	igual = 1	
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	
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	
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	
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	
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	
16	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000,	igual = 1	

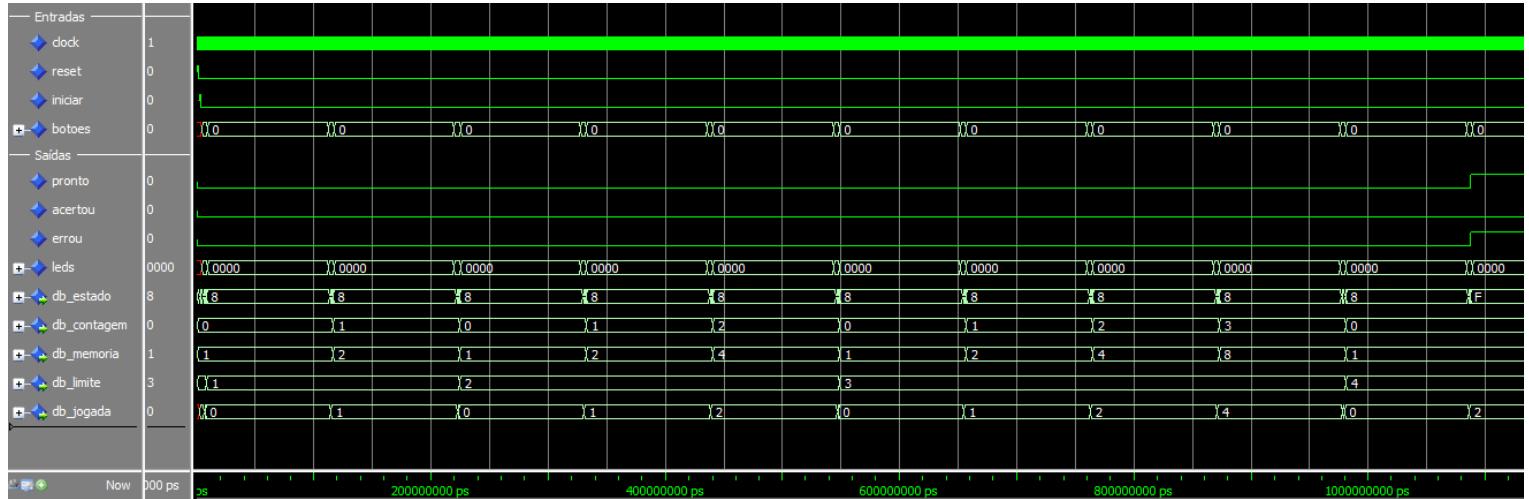
		botoes = 0100, botoes = 0010, botoes = 0001, botoes = 0001, botoes = 0010, botoes = 0010, botoes = 0100, botoes = 0100, botoes = 1000, botoes = 1000, botoes = 0001		
17	Acertar jogada	botoes = 0001, botoes = 0010, botoes = 0100, botoes = 1000, botoes = 0100, botoes = 0010, botoes = 0001, botoes = 0001, botoes = 0010, botoes = 0010, botoes = 0100, botoes = 0100, botoes = 1000, botoes = 1000, botoes = 0001, botoes = 0100	igual = 1, pronto = 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	
3		botoes = 0001, botoes = 0010	igual = 1	
4		botoes = 0001, botoes = 0010, botoes = 0100	igual = 1	
5	Errar jogada	botoes = 0010	pronto = 1, errou = 1	

2.4) Simulação do circuito



Nesta simulação, realizamos primeiro o cenário de testes #2, em seguida o cenário de testes #1 e por fim, novamente o cenário de testes #1.



Zoom no caso de testes #2. Aqui é possível ver com mais detalhes a mudança nos sinais de depuração, que indicam o funcionamento correto do circuito.

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

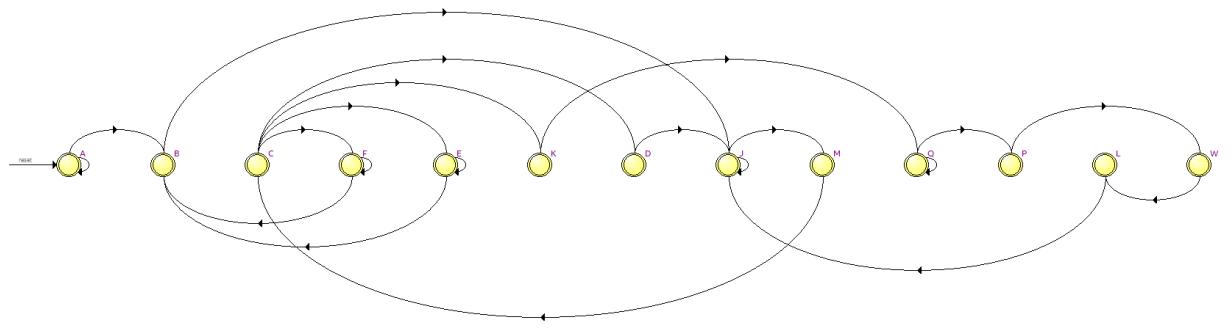
RESET	GPIO_0_D15	PIN_N19	StaticIO – Button 0/1 – DIO1
INICIAR	GPIO_0_D17	PIN_P19	StaticIO – Button 0/1 – DIO2
BOTOES(0)	GPIO_0_D19	PIN_P17	StaticIO – Button 0/1 – DIO3
BOTOES(1)	GPIO_0_D21	PIN_M18	StaticIO – Button 0/1 – DIO4
BOTOES(2)	GPIO_0_D23	PIN_L17	StaticIO – Button 0/1 – DIO5
BOTOES(3)	GPIO_0_D25	PIN_K17	StaticIO – Button 0/1 – DIO6
LEDS(0)	Led LEDR0	PIN_AA2	-
LEDS(1)	Led LEDR1	PIN_AA1	-
LEDS(2)	Led LEDR2	PIN_W2	-
LEDS(3)	Led LEDR3	PIN_Y3	-
PERDEU	Led LEDR7	PIN_U1	-
GANHOU	Led LEDR8	PIN_L2	-
PRONTO	Led LEDR9	PIN_L1	-
db_chavesIgualMemoria	Led LEDR4	PIN_N2	-
db_clock	Led LEDR5	PIN_N1	-
db_tem_jogada	Led LEDR6	PIN_U2	-
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	-

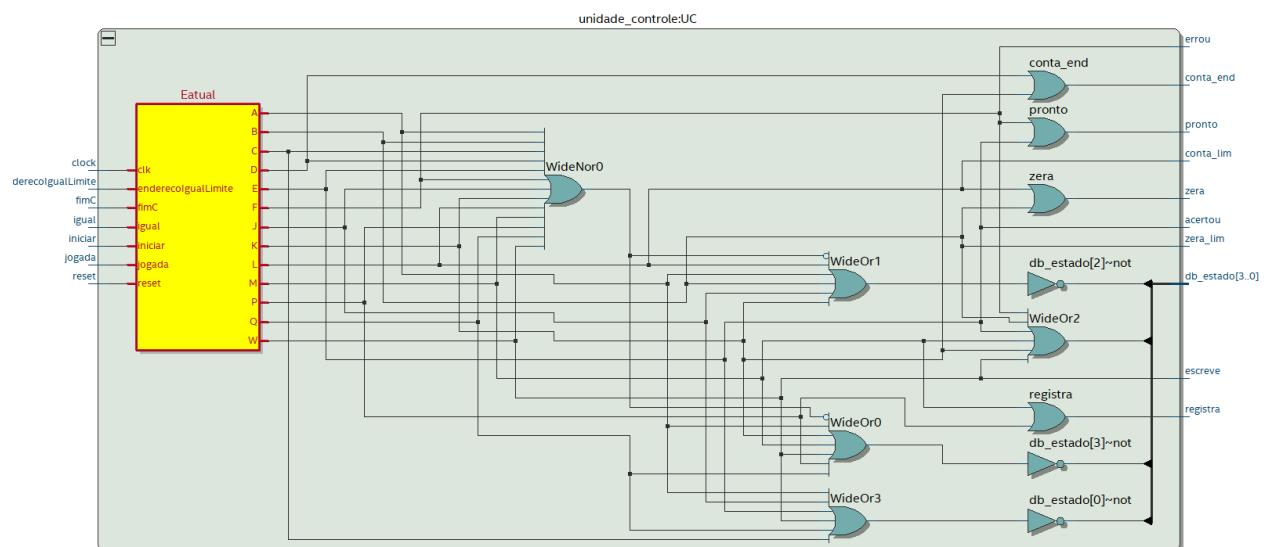
3) Descrição do Desafio

O desafio segue uma lógica similar ao circuito original, mas com um adicional de gravar a próxima jogada a cada turno. Por exemplo, na primeira jogada, o jogador deve acertar o primeiro valor (0001, na posição 0000 da memória) na memória, e em seguida gravar a próxima jogada, através dos botões (Por exemplo, se ele apertar o botão 2, será gravado o valor 0100 na posição 0001 da memória). No próximo turno, ele deve acertar a primeira jogada e depois aquela que foi gravada, em seguida gravar a terceira jogada. E assim por diante até chegar ao fim da memória. A partir daí ele tem que acertar todas as jogadas gravadas. Se ele errar qualquer jogada ao longo do funcionamento, o circuito vai para o estado de erro. Se ele acertar todas as jogadas em todos os turnos ele vai para o de acerto. Essa adição do funcionamento da memória foi feita adicionando-se 4 estados à unidade de controle, e ligando a entrada de Write Enable da memória à unidade de controle. A memória recebe o dado de entrada do registrador de jogadas. Os estados adicionais foram K,Q,P e W. O estado K vem logo depois do estado de comparação, onde se verifica se o contador de endereços se igualou ao de limite. Se ele não se igualou, o circuito segue para o estado D onde incrementa o contador de endereços e continua as jogadas. Se ele se igualou, quer dizer que chegou no limite, ou seja, ele vai para o estado K onde ele prepara para a gravação incrementando um no contador de endereços. Em seguida, ele vai para o estado Q, onde espera um acionamento dos botões. Ao receber um acionamento dos botões ele vai para o estado P, onde apenas aguarda um ciclo de clock, para que a jogada seja armazenada no registrador. Em seguida vai para o estado W, onde aciona a entrada Write Enable da memória, guardando assim a próxima jogada. Depois, vai para o estado L, onde incrementa o limite, e zera o contador de endereços.

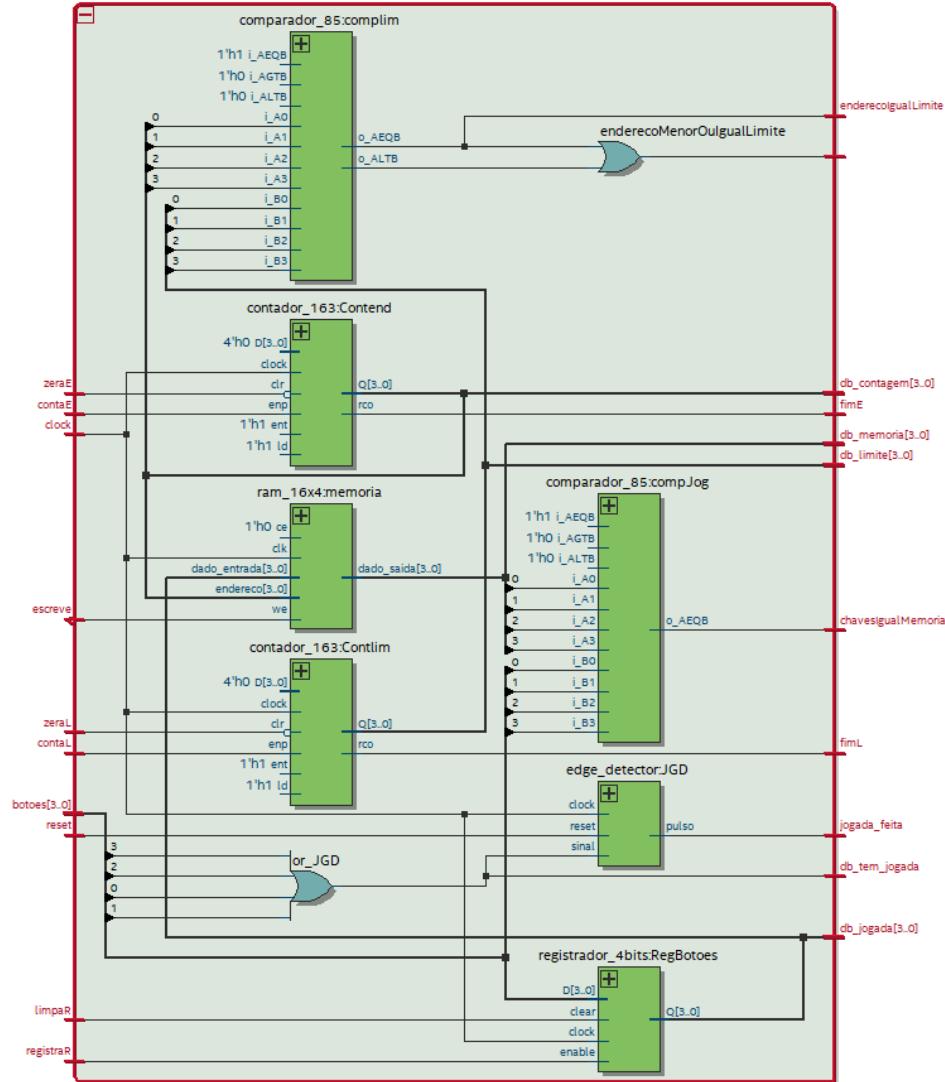
Descrição da máquina de estados:



Unidade de controle:

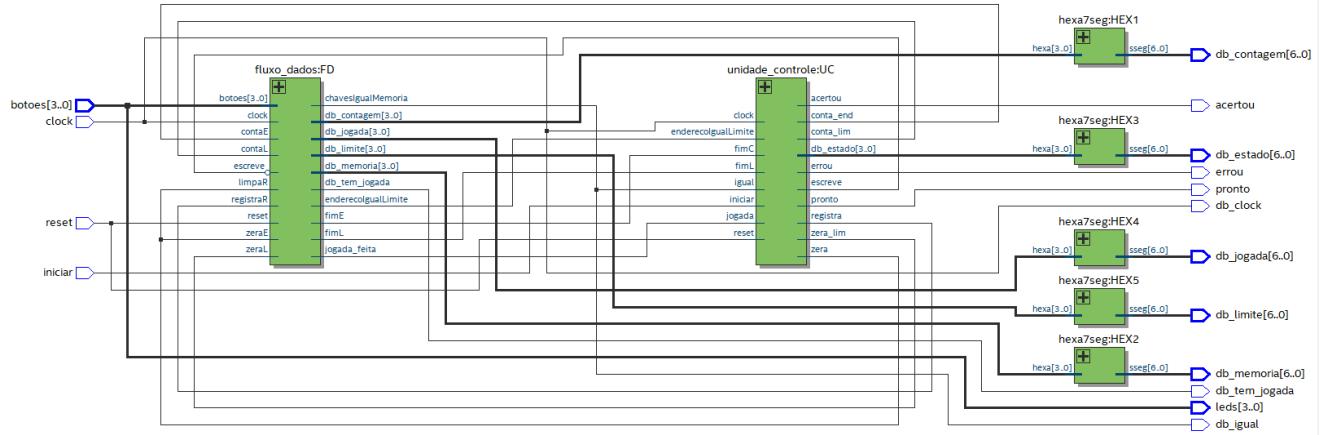


Fluxo de dados:



Destaque para a entrada Escreve.

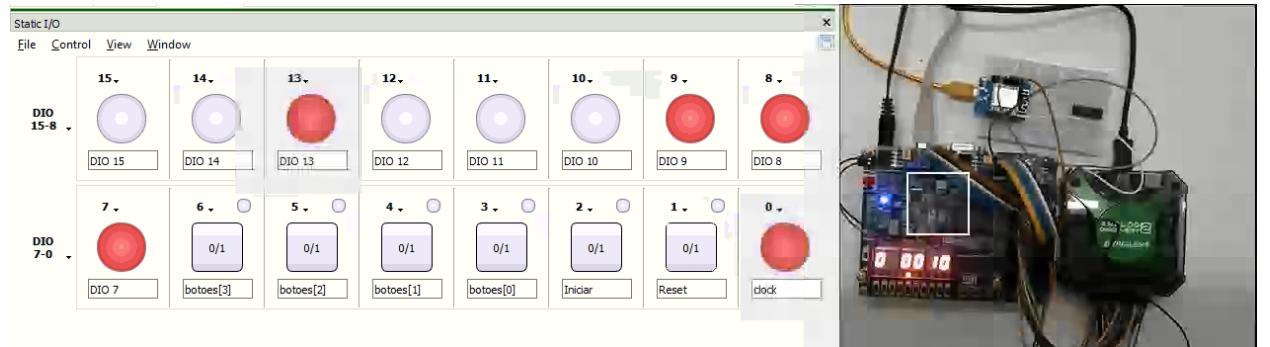
Circuito geral:



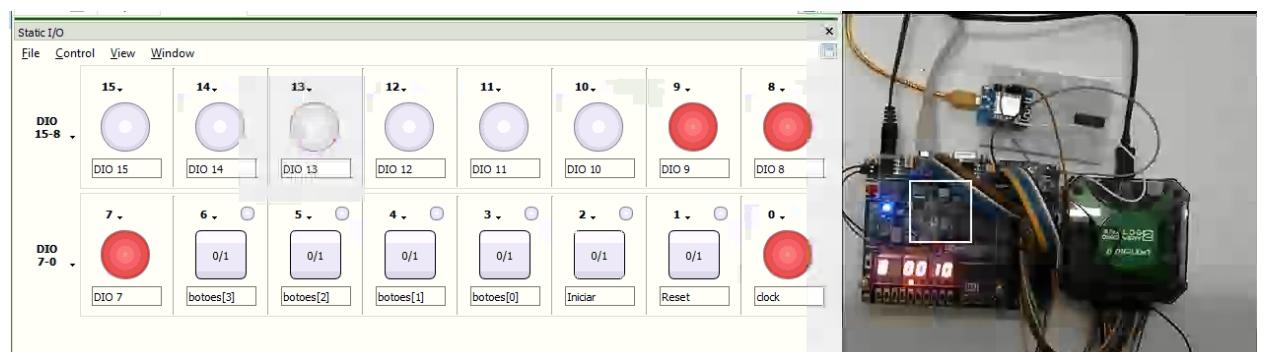
4) Demonstração do Funcionamento do Circuito

3.1) Acerto de todas as jogadas

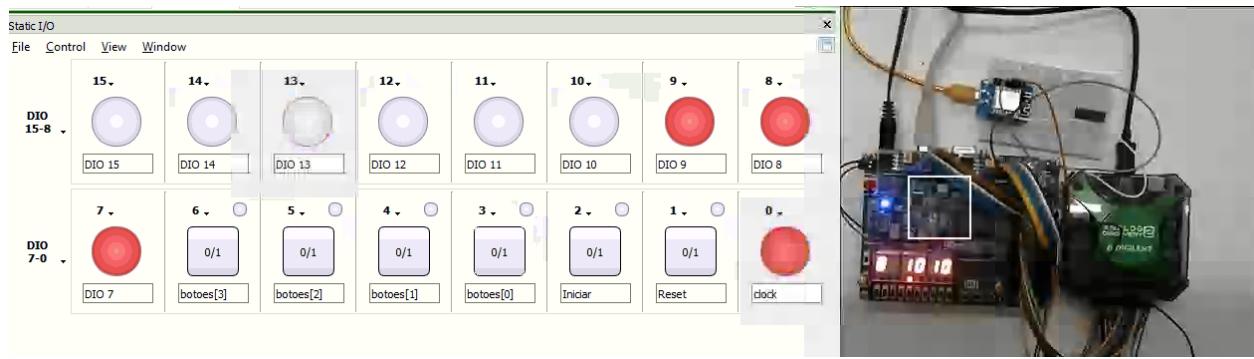
3.1.1) Início do ciclo



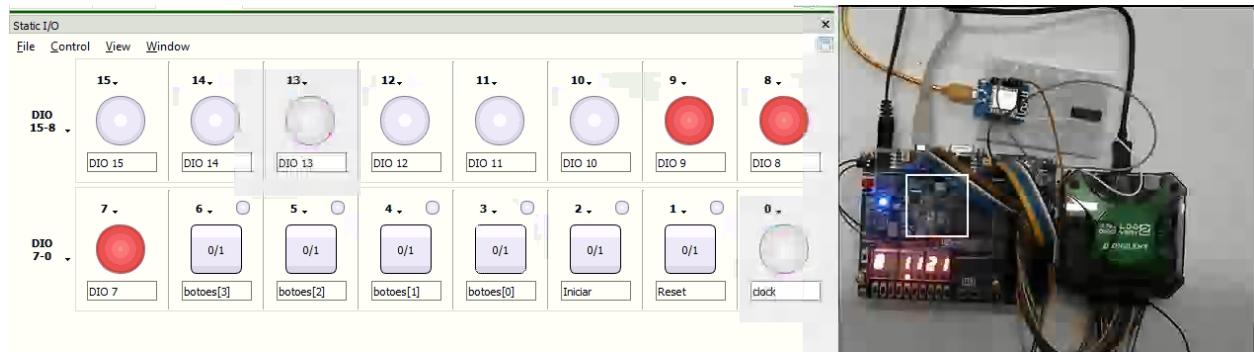
Condições iniciais



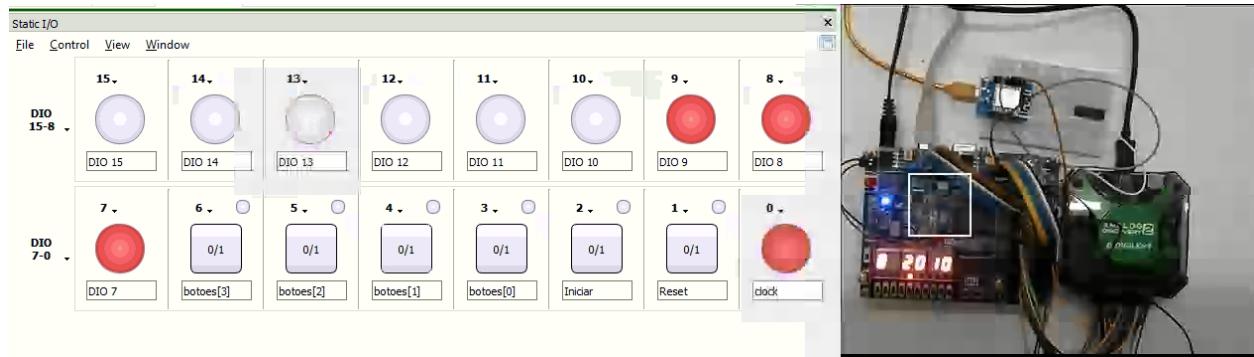
Após apertar o botão “iniciar” uma vez, o ciclo de jogadas se inicia



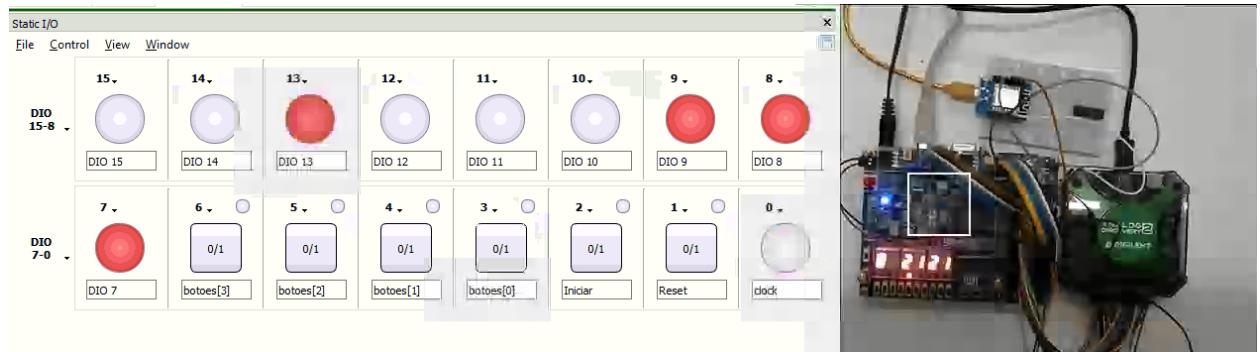
Acerto da primeira rodada (0001)



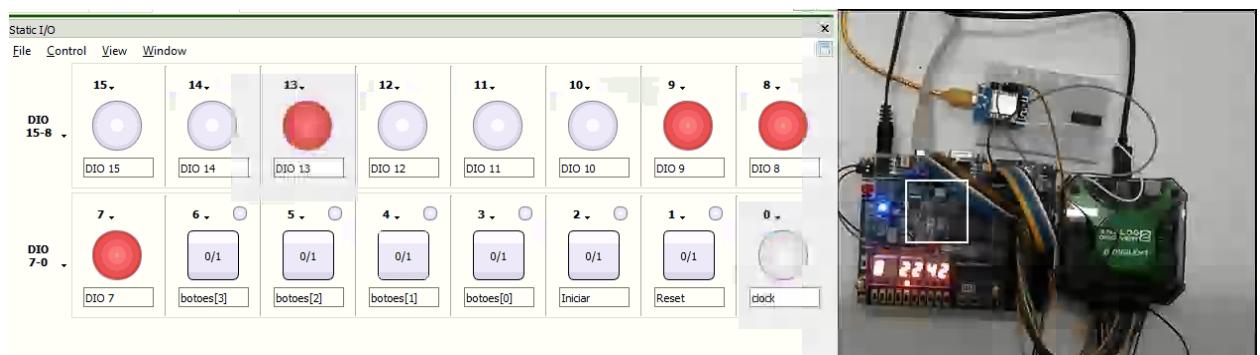
Acerto da primeira jogada da segunda rodada (0001)



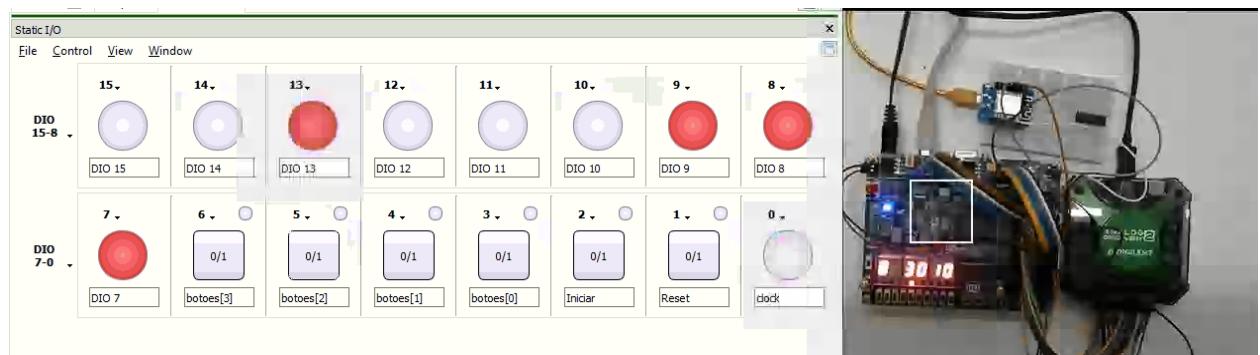
Acerto da segunda e última jogada da segunda rodada (0010)



Acerto da primeira jogada da terceira rodada (0001)

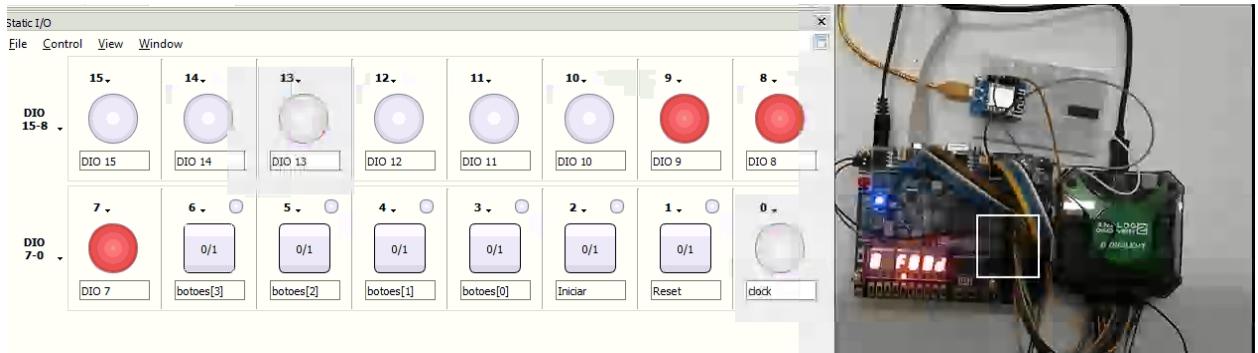


Acerto da segunda jogada da terceira rodada (0010)

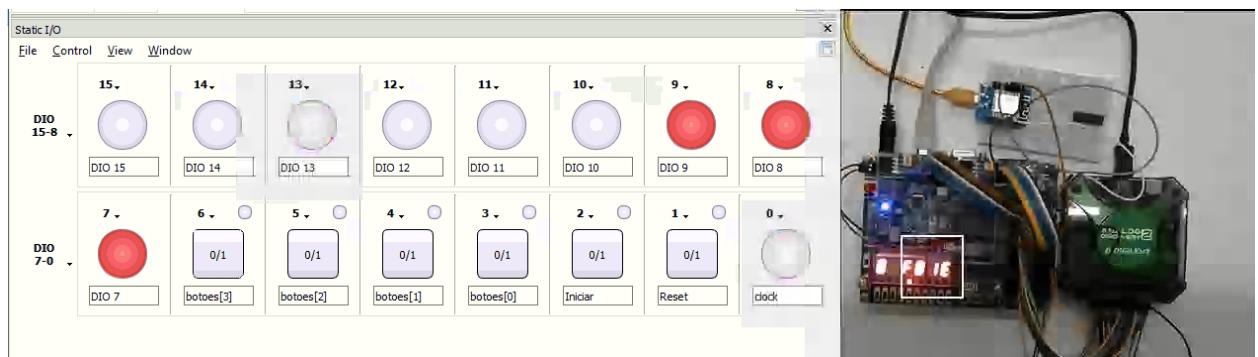


Acerto da terceira e última jogada da terceira rodada (0100)

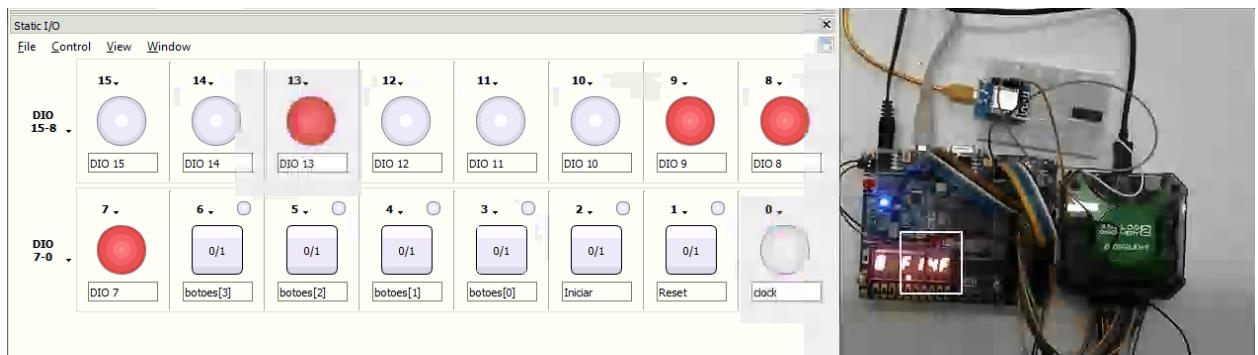
3.1.2) Fim do ciclo



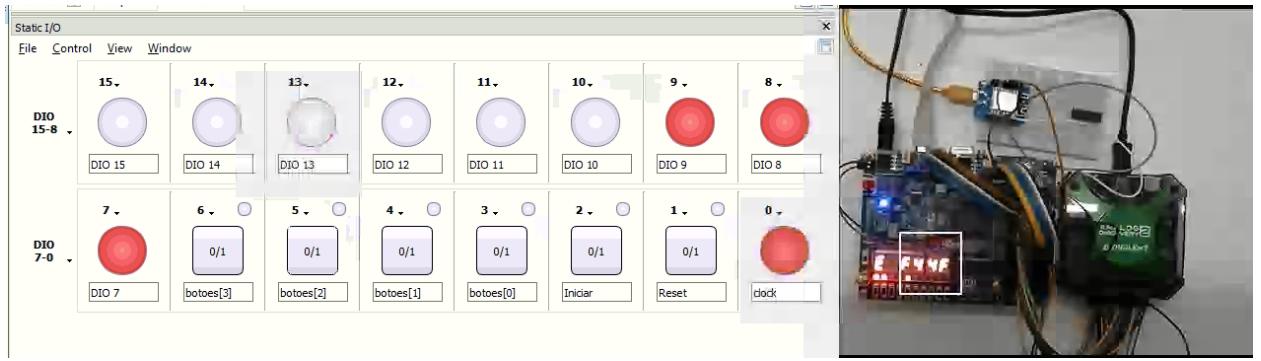
Neste caso, o circuito encontra-se próximo ao fim do ciclo de rodadas. Acerto da 13^a jogada da 16^a rodada (1000)



Acerto da 14^a jogada da 16^a rodada (1000)

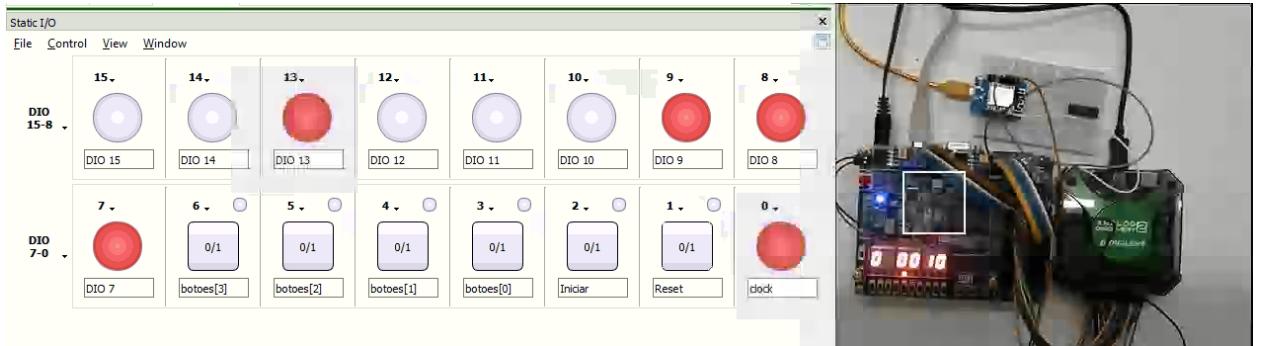


Acerto da 15^a jogada da 16^a rodada (0001)

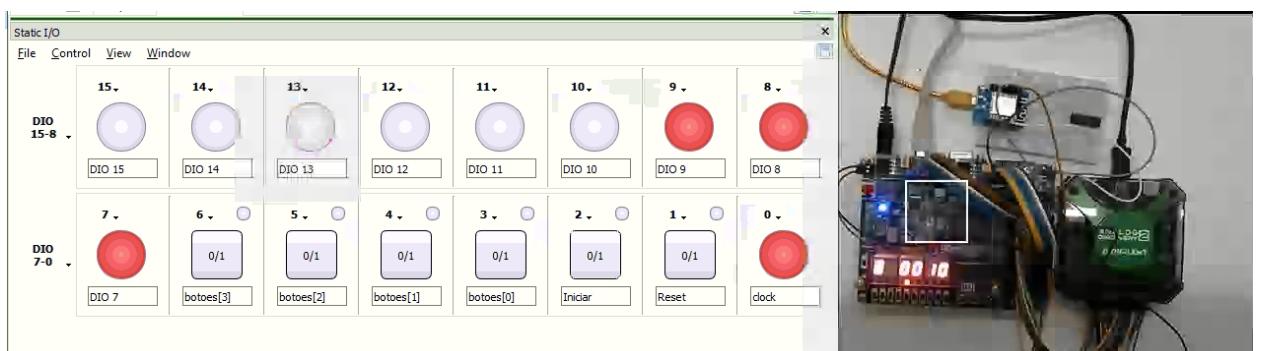


Acerto da 16^a e última jogada da 16^a rodada (0100). O circuito vai para o estado que indica que todas as rodadas foram acertadas (E) e os LEDs ligados aos sinais “ganhou” e “pronto” acendem.

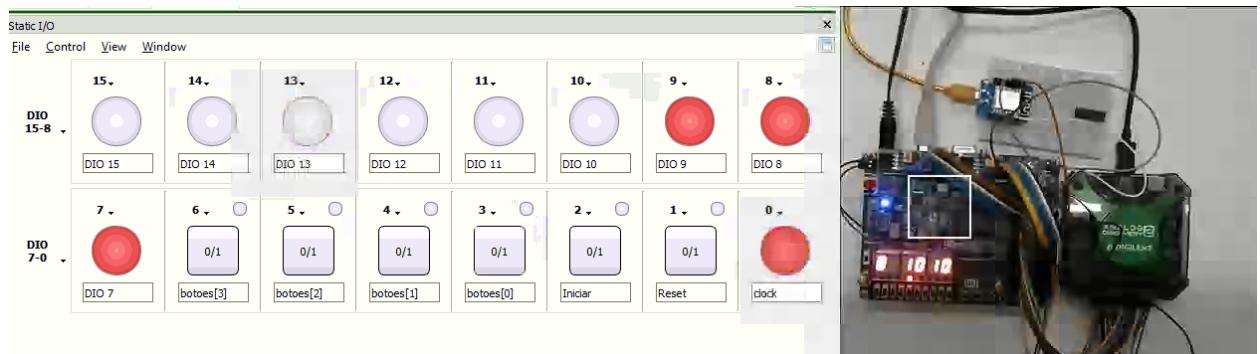
3.2) Erro em uma jogada



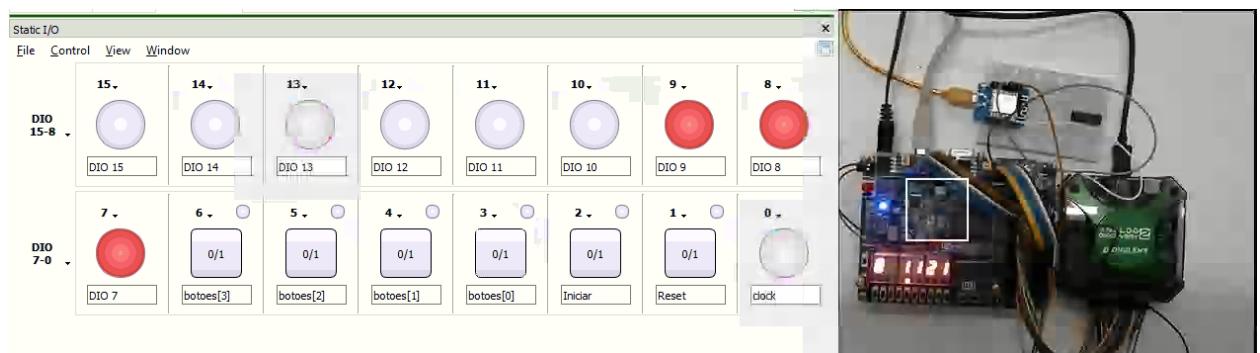
Condições iniciais



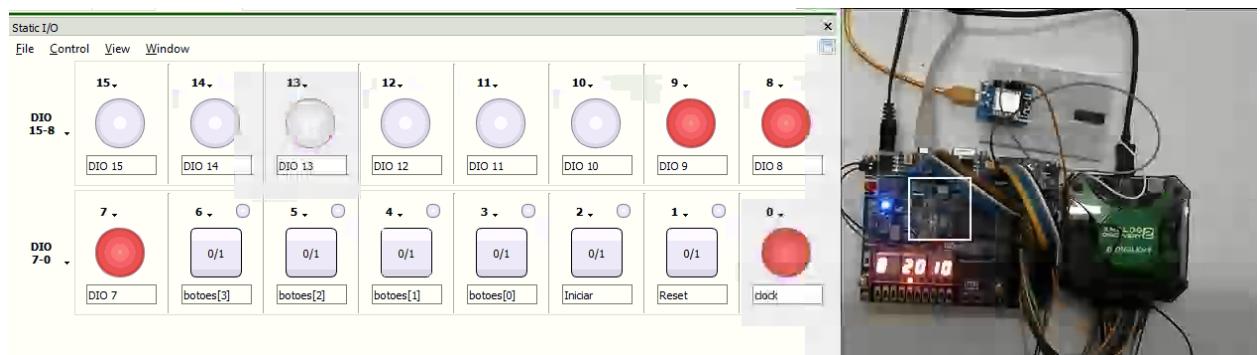
Após apertar o botão “iniciar” uma vez, o ciclo de jogadas se inicia



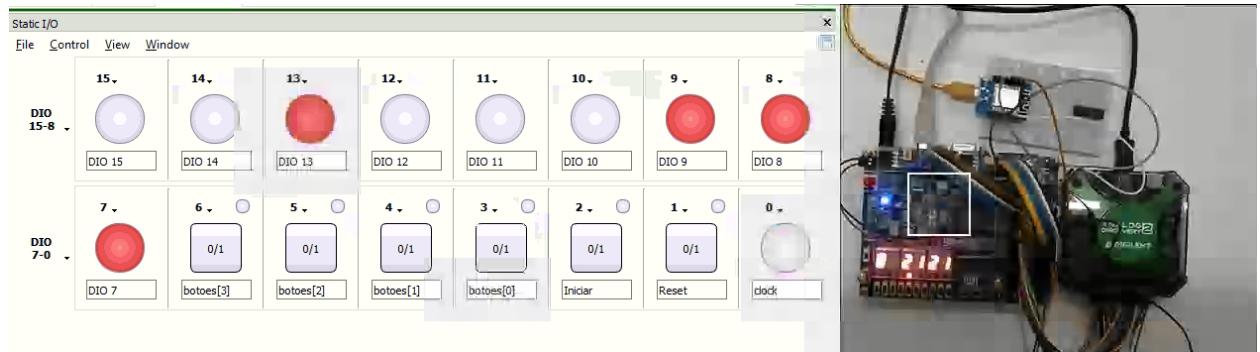
Acerto da primeira rodada (0001)



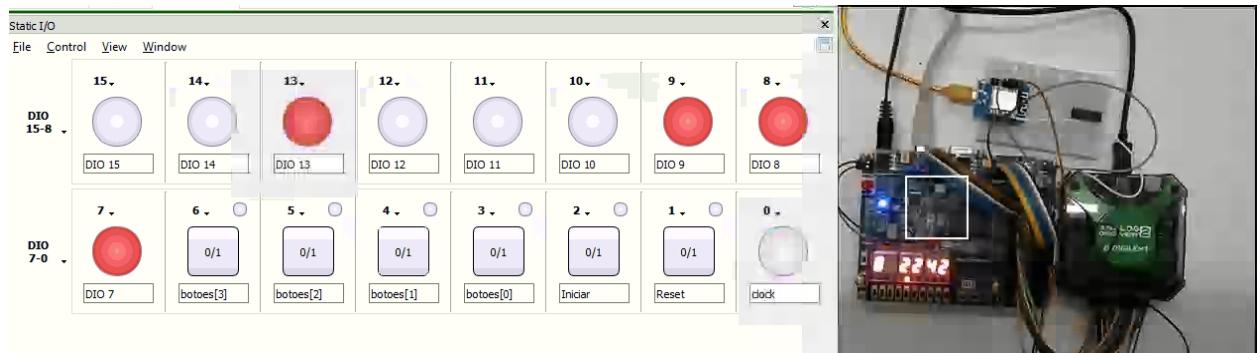
Acerto da primeira jogada da segunda rodada (0001)



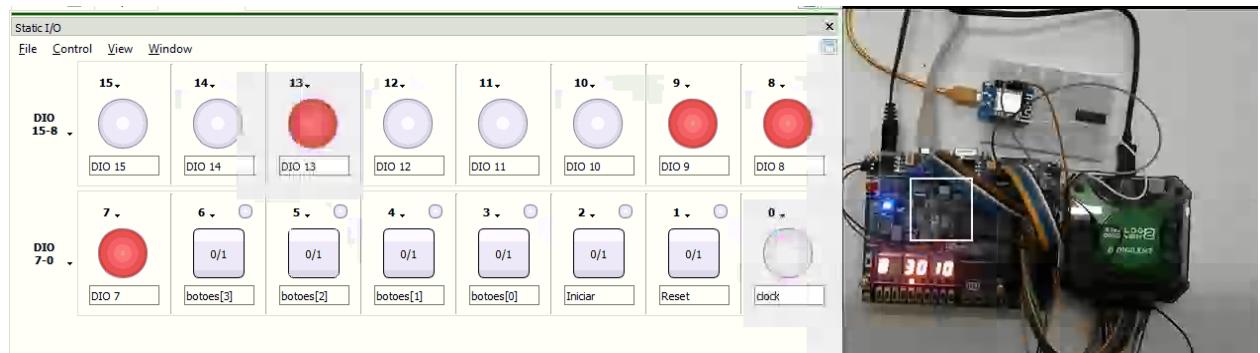
Acerto da segunda e última jogada da segunda rodada (0010)



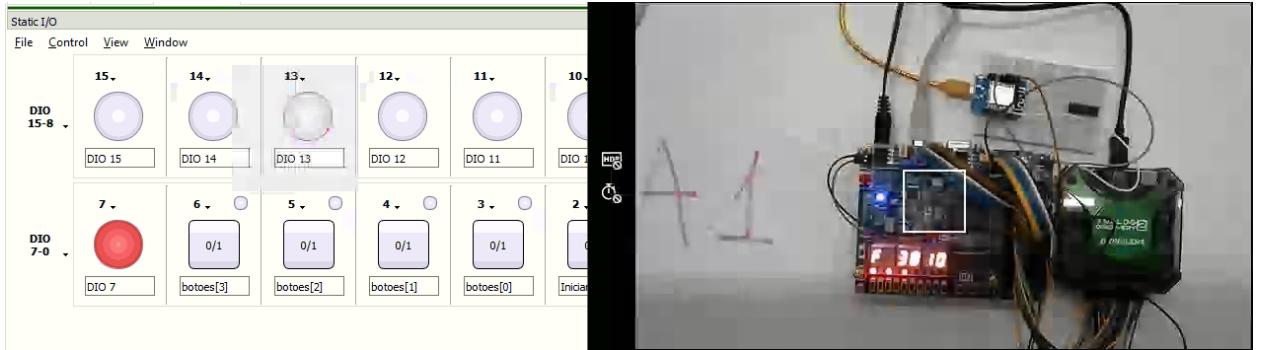
Acerto da primeira jogada da terceira rodada (0001)



Acerto da segunda jogada da terceira rodada (0010)



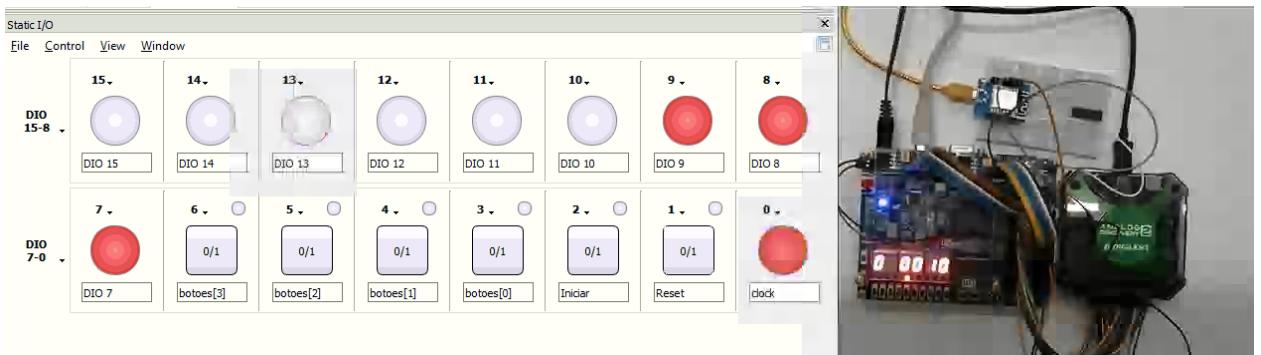
Acerto da terceira e última jogada da terceira rodada (0100)



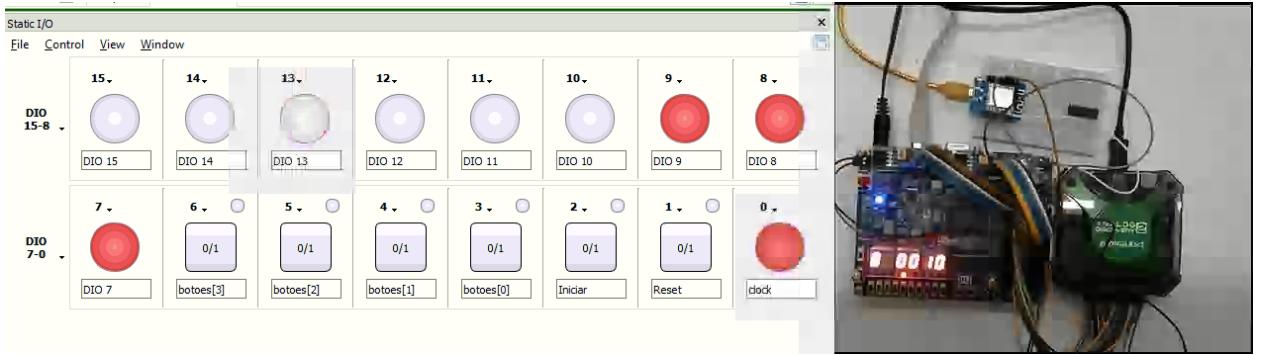
Erro da primeira jogada da quarta rodada (valor esperado: 0001, valor inserido: 1000). O circuito vai para o estado que indica que uma jogada foi incorretas (F) e os LEDs ligados aos sinais “perdeu” e “pronto” acendem.

3.3) Desafio

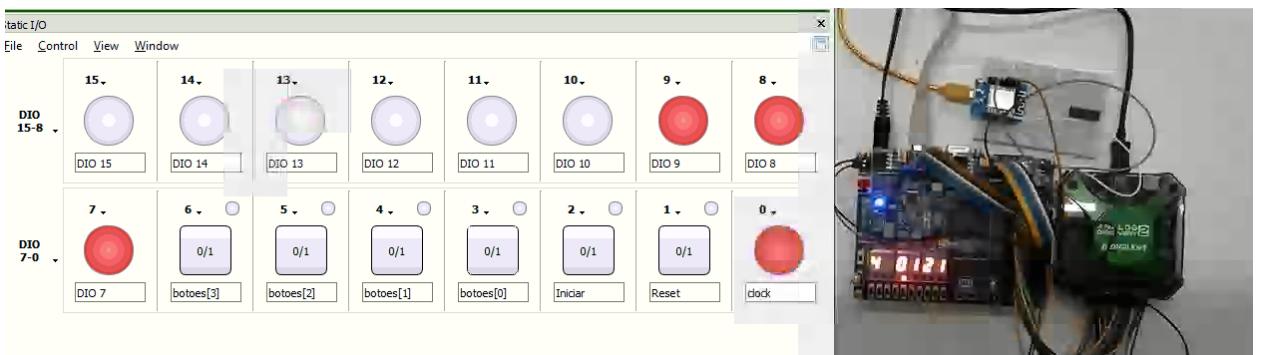
Na depuração do circuito foi alocado, da direita para a esquerda, os visores de 7 segmentos para o contador de endereços, o dado da memória, a jogada anterior, o contador de limite e o estado atual. Os estados são representados como A=0(inicial), B=B(Início), J=8(espera jogada), M=7(atualiza registrador), C=C(compara), D=D(incrementa contador de endereços), E=E(acerto), F=F(erro), L=9 (Incrementa limite e zera o endereço), K=3 (incrementa endereço a mais), W=6 (escreve na memória), Q=4 (espera jogada a ser escrita), P=5 (atualiza registrador)



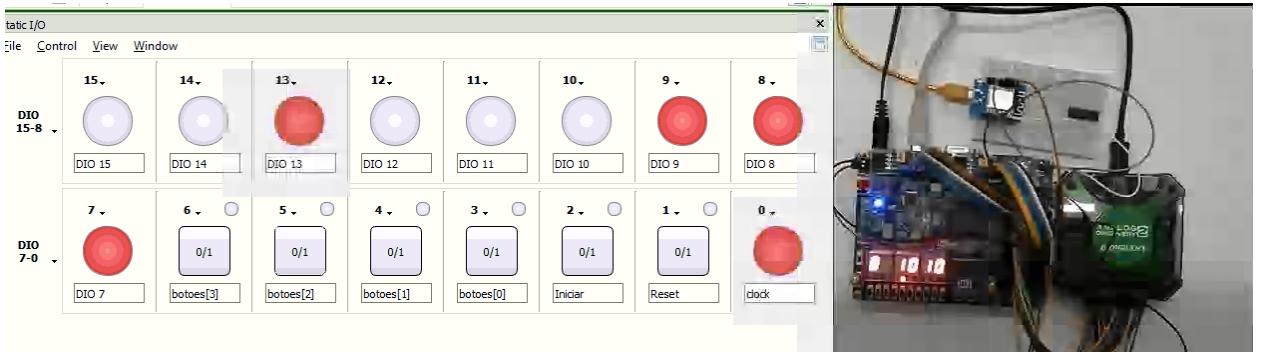
Situação Inicial: estado A(0), memória e limite na posição inicial. Conteúdo da memória =0001 na posição 0000



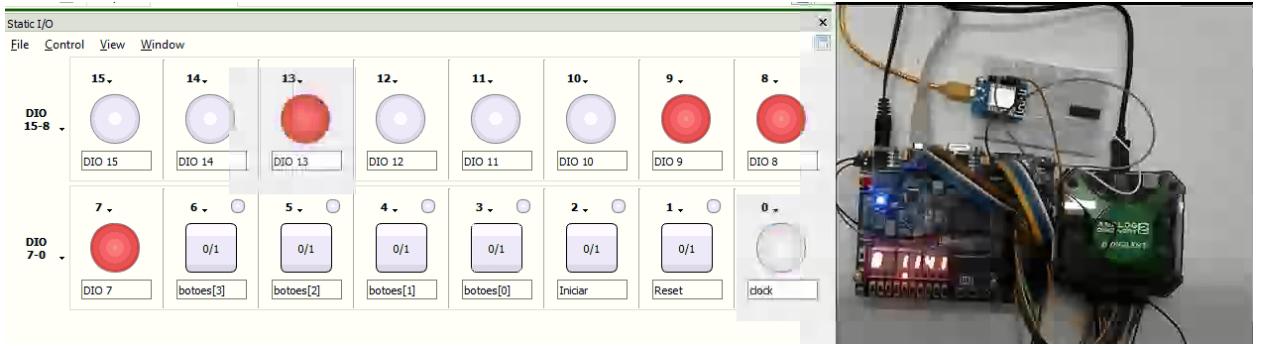
Iniciar acionado, o circuito passou pelo estado B e está no estado J, esperando uma jogada.



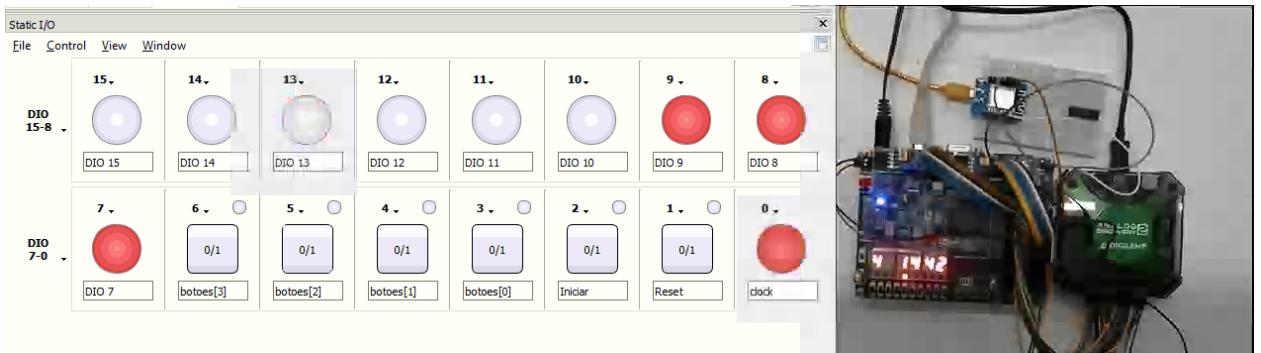
Jogou-se 0001, o que está correto. O circuito está com o mesmo valor no contador limite e endereço, logo passa pela sequência de salvar na memória. O circuito passa pelo estado K, incrementando mais 1 no endereço e está no estado Q, esperando uma jogada a ser salva no endereço 0001 da memória.



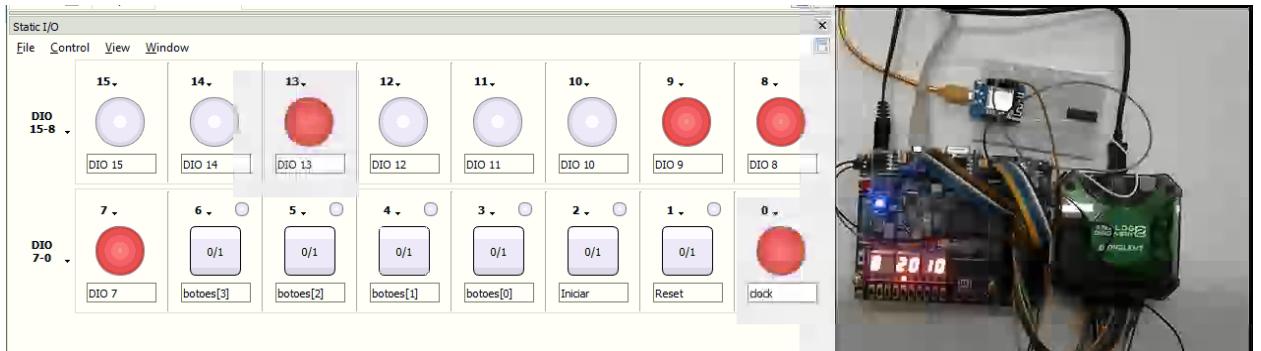
Após realizada a jogada salva, o circuito passa pelo estado P para registrar a jogada, e depois pelo estado W para escrever a jogada na memória. Nesse caso foi salvo a jogada 0100. Em seguida, ele passa pelo estado L, onde incrementa o limite e espera uma jogada no estado J. Nota-se que na foto ele se encontra no estado J e está com o contador de limite contando 0001 agora, e o contador de endereços na posição 0000, com a memória armazenando 0001.



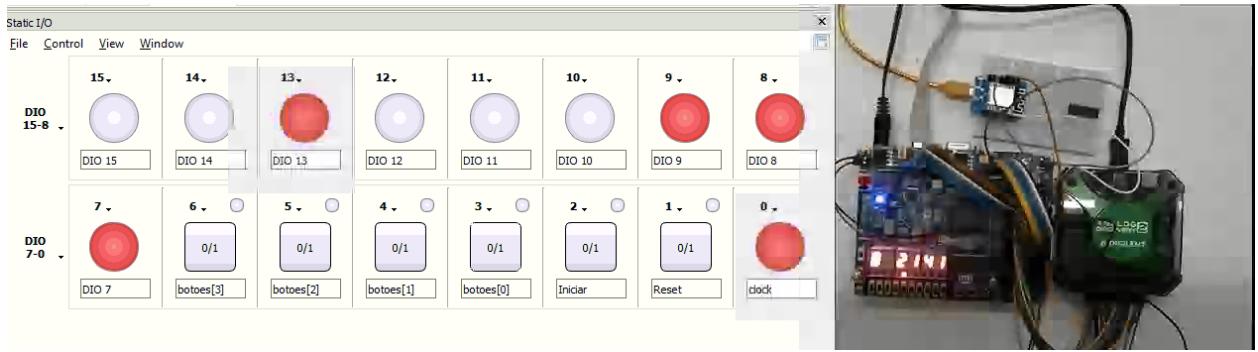
Após realizada uma jogada correta, o circuito passa pelo estado M de registrar a jogada, e vai para o estado de comparação C. Como o endereço não é igual ao limite, o circuito vai para o estado D, onde incrementa o contador de endereços uma vez, e espera uma jogada no estado J. Na foto, repara-se que o conteúdo na memória foi de fato alterado na rodada anterior, pois o endereço 0001 agora guarda 0100 (a jogada salva) ao invés de 0010.



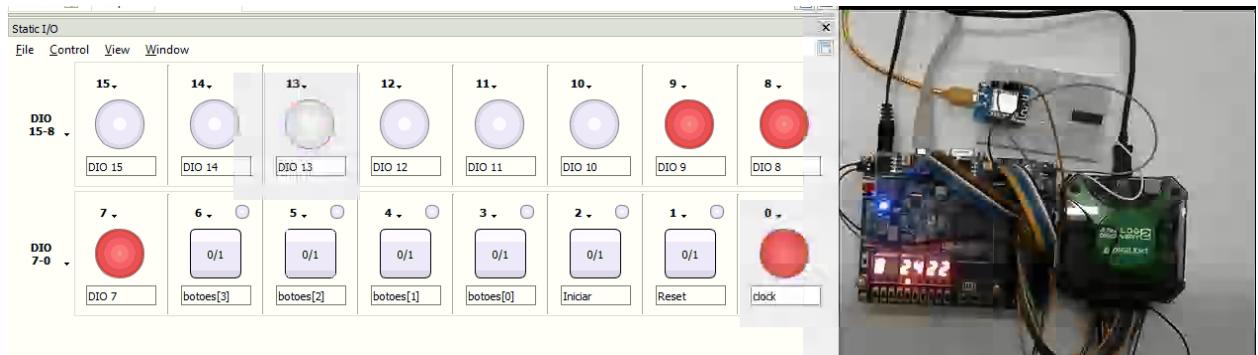
Ao se fazer a jogada certa (0100), o circuito passa de volta à lógica de gravar a próxima jogada. Na foto, está no estado Q esperando a jogada a ser salva e irá guardar a jogada no endereço 0010.



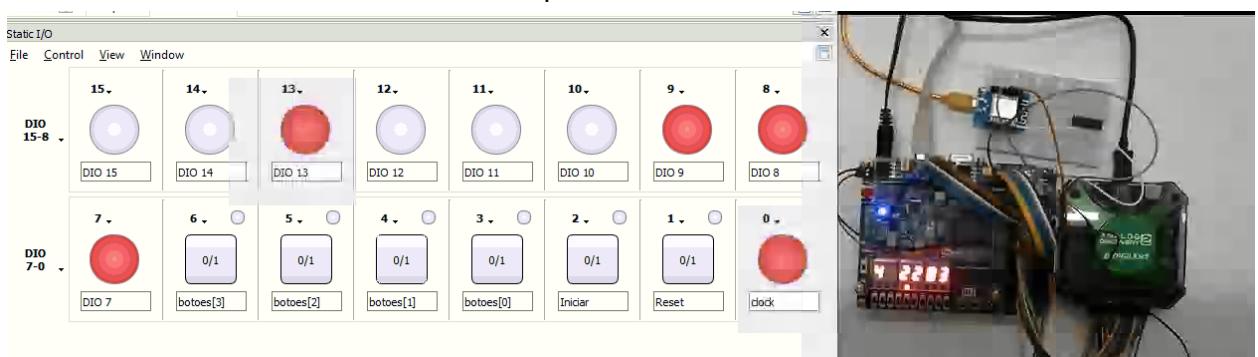
Armazenada a jogada (0010), ele incrementa o limite e zera o endereço. Espera uma jogada no J.



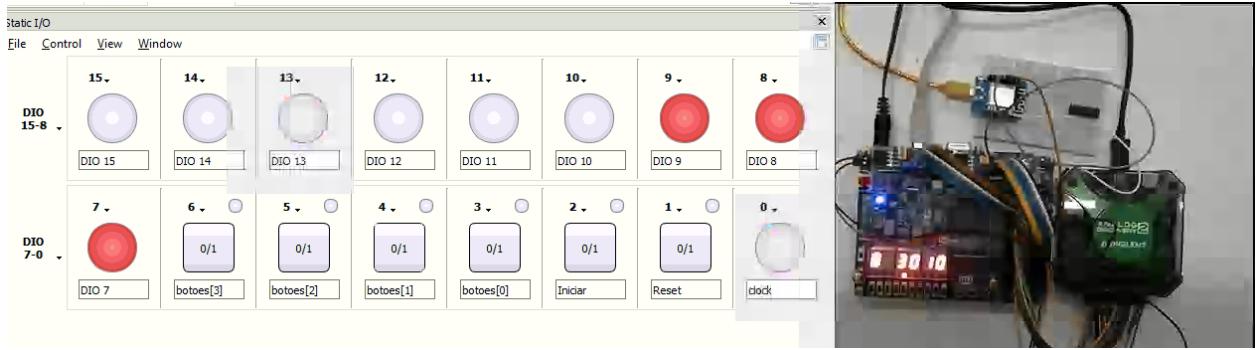
Realizada a jogada correta no endereço 0001 (0001), o circuito incrementa 1 no contador de endereços e aguarda uma jogada. Nota-se novamente que o circuito ainda guarda 0100 no endereço 0001.



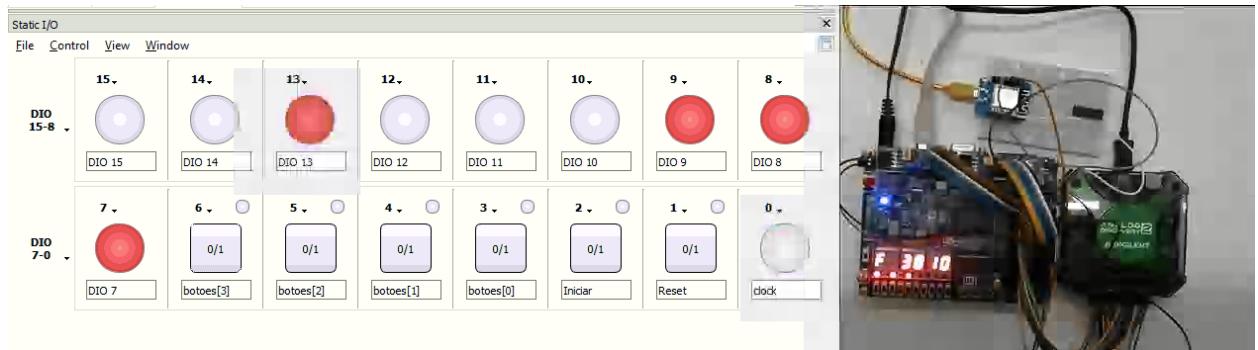
Realizada a jogada correta 0100, o circuito agora incrementa mais um no contador de endereços e aguarda a próxima jogada. Nota-se que no endereço 2 está guardado o valor 0010 ao invés de 0100, mostrando que a escrita foi um sucesso.



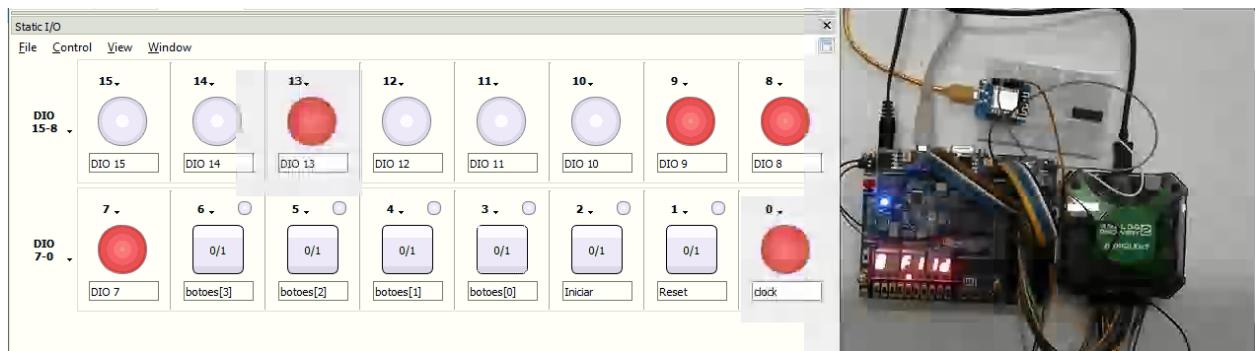
Como o endereço é igual ao limite, ele incrementa 1 endereço e aguarda uma jogada a ser salva nele.



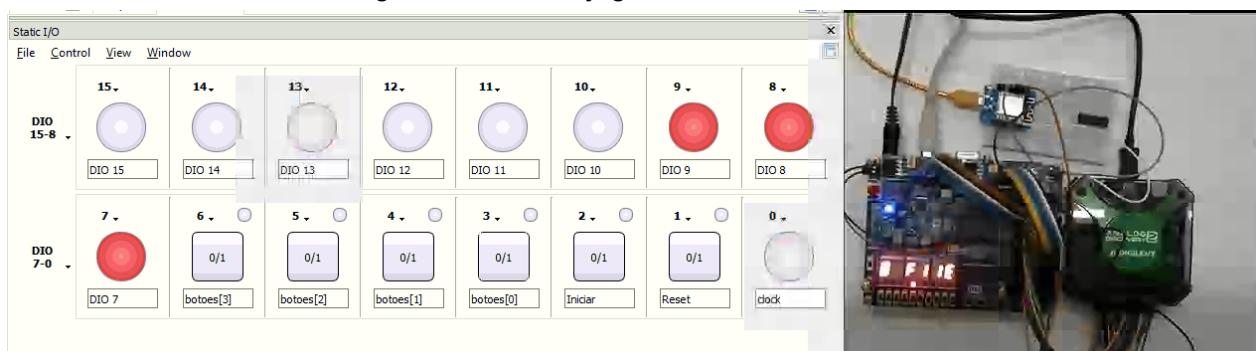
Salva a jogada, ele zera os endereços e incrementa 1 no limite.



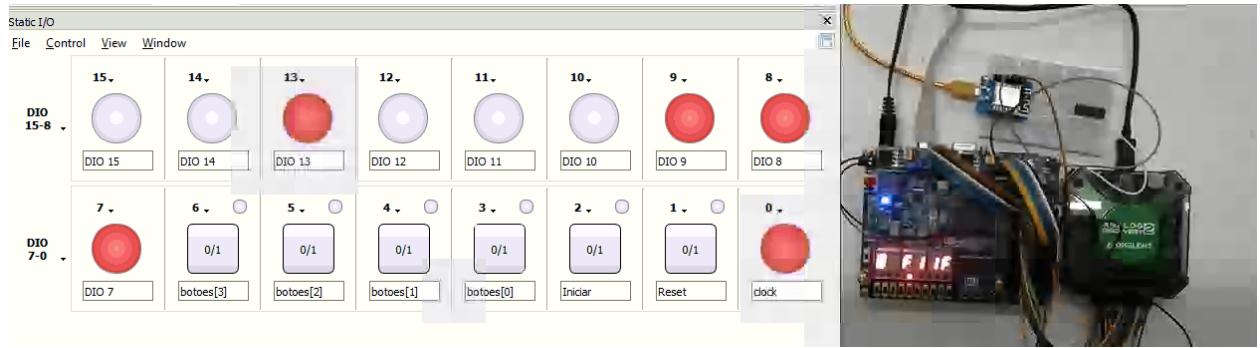
Se realizada uma jogada errada, o circuito vai para o estado de erro F.



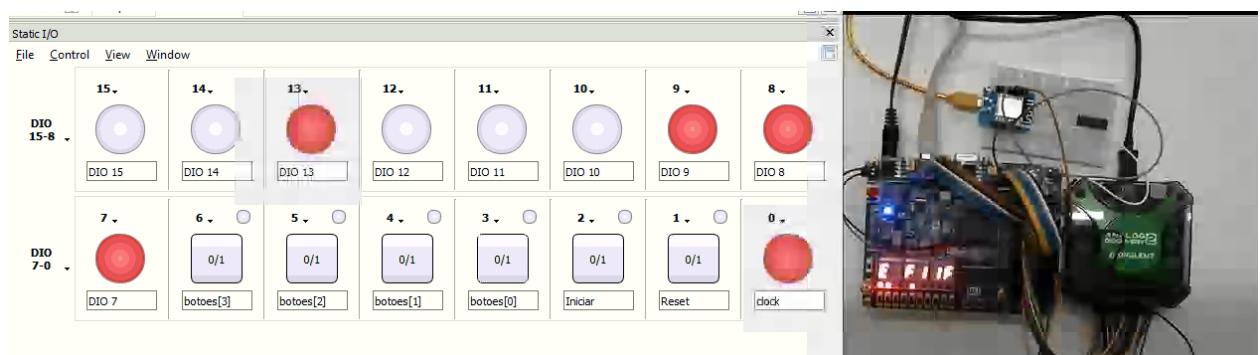
Após o teste de erro, o circuito foi reiniciado apertando-se o botão “iniciar” novamente e para o teste do acerto foi-se colocada uma sequência de 0001 em todas as jogadas até se chegar na última rodada, onde deve-se acertar todo o conteúdo da memória. O circuito encontra-se com o limite no máximo F, e no endereço D, onde está salvo 0001. O circuito está no estado J aguardando uma jogada



Mesma situação, porém foi incrementado 1 no endereço pela jogada correta



Agora na última jogada, nota-se que está salvo 1 também no último endereço da memória.



Ao se acertar o conteúdo do último endereço o circuito vai para o estado de acerto.

5) Resultados Alcançados

5.1) Pontos Positivos

Apesar das dificuldades encontradas na elaboração do planejamento e no início do desenvolvimento da atividade no laboratório, a experiência correu bem e conseguimos cumprir todos os objetivos pretendidos dentro do prazo esperado, implementando devidamente os circuitos “circuito_exp6” e “circuito_exp6_desafio”.

5.2) Pontos Negativos

Um dos pontos negativos desse experimento foi no planejamento. Nos organizamos mal para preparar o planejamento e o desenvolvimento foi meio apressado. Conseguimos montar o circuito, mas ele estava falhando em todos os testes e não identificamos o porquê. Como já estava tarde e a entrega era pro dia seguinte optamos por entregar do jeito que estava e pensar no problema no dia seguinte com a mente descansada. No dia seguinte, identificamos rapidamente o problema no testbench, que estava realizando as operações de forma dessincronizada e causava o erro no circuito (acionava os botões fora da hora). Ao se arrumar o testbench vimos que o circuito funcionava corretamente, pelo menos em teoria. No dia do laboratório estávamos tendo um problema em

rodar o circuito em clocks altos. Se fosse rodado o circuito igual no testbench, com o clock em 1 hz, realizando passo a passo as operações, ele funcionava normalmente, se aumentássemos o clock para khz ele parava de funcionar corretamente. Identificamos junto com o monitor o problema na configuração do detector de borda, que estava sendo resetado junto com o contador de endereços ao invés de junto com o circuito geral. Após arrumar isso ele operou corretamente.

5.3) Lições Aprendidas

As lições aprendidas novamente foram ligadas a importância de um bom planejamento. Para o próximo experimento vamos nos organizar melhor quanto ao tempo para fazermos o planejamento e assim podemos fazê-lo com calma. Além disso, reorganizamos os arquivos VHDL para serem menores e mais distribuídos, ao invés de guardar tudo em um arquivo só. Isso ajuda demais na manutenção do projeto, ainda mais que o circuito está ficando cada vez mais complexo, e é uma boa prática que aprendemos a adotar. Além disso, aprendemos bem a relevância do testbench, que facilita na hora de testar o circuito, ainda mais agora que para se chegar no acerto são necessárias mais de 150 acionamento dos botões. O experimento foi um bom aprendizado para as dificuldades de um projeto mais complexo, onde os problemas podem vir de vários componentes diferentes, o que torna a documentação e a organização essenciais para uma boa prática de engenharia.

6) Referências Técnicas

1. Apostila e apresentação da Experiência 6, disponibilizados no ambiente da disciplina PCS3635 no site E-disciplinas.
2. Tabela de Pinos da placa DE0-CV