



**Laboratório Digital I - PCS3635**

**Relatório da Experiência 3:**

**Desenvolvendo um Fluxo de Dados**

Marco Aurélio C. O. Prado - NUSP 11257605

Victor Hoefling Padula - NUSP 10770051

Turma 04 - Bancada A1

São Paulo - SP

22/01/2021

## 1) Objetivo

Os objetivos da aula consistem em aprender sobre:

- Projeto de circuitos usando descrição estrutural VHDL;
- Uso de um módulo de memória;
- Interface com display de 7 segmentos;
- Documentação de projetos (planejamento e relatório);
- Estudo de um fluxo de dados de um circuito digital.

## 2) Estudo da descrição VHDL fornecida

### 2.1) Descrição VHDL comentada

---

```
-- Arquivo : ram_16x4.vhd
-- Projeto : Experiencia 03 - Desenvolvendo o Fluxo de Dados
```

---

```
-- Descricao : módulo de memória RAM 16x4
--           sinais we e ce ativos em baixo
--           código ADAPTADO do código encontrado no livro
--           VHDL Descricao e Sintese de Circuitos Digitais
--           Roberto D'Amore, LTC Editora.
```

---

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

---

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

entity ram_16x4 is -- declaracao da entidade
    port ( -- entradas
        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) -- saida
    );
end entity ram_16x4;
```

```
architecture ram1 of ram_16x4 is -- inicio da arquitetura da memoria
```

```

type arranjo_memoria is array(0 to 15) of std_logic_vector(3 downto 0);
-- array de 16 posicoes de vetores de 4 bits
signal memoria : arranjo_memoria;
attribute ram_init_file: string;
attribute ram_init_file of memoria: signal is "ram_conteudo_inicial.mif";
-- arquivo mif de inicializacao da memoria
begin

process(ce, we, endereco, memoria) -- process que define o funcionamento
da memoria
begin
    if ce = '0' then -- dado armazenado na subida de "we" com "ce=0"
        if rising_edge(we)
            then memoria(to_integer(unsigned(endereco))) <= dado_entrada; --
escrita de “dado_entrada” na posicao de memoria “endereco”
        end if;
        end if;
        dado_saida <= memoria(to_integer(unsigned(endereco))); -- leitura do
dado armazenado na posicao de memoria “endereco”
    end process;

end architecture ram1; -- fim da arquitetura

```

## 2.2) Perguntas a respeito do circuito fornecido

### 2.2.1) Mostre a sequência de sinais para um ciclo de leitura de memória.

endereco: endereço de memória que se deseja ler  
 dado\_entrada: x  
 we: x  
 ce: x

### 2.2.2) Mostre a sequência de sinais para um ciclo de escrita de memória.

endereco: endereço de memória onde se deseja escrever  
 dado\_entrada: dado a ser escrito  
 we: 1

ce: 0

### 2.2.3) Explique o conteúdo de um arquivo MIF.

Um arquivo MIF(*Memory Initialization File*) tem como função inicializar a memória, ou seja, preenchê-la com valores prévios antes do circuito começar a funcionar.

Dentro deste arquivo, temos como parâmetros:

DEPTH = 4; -- A quantidade de palavras da memória  
WIDTH = 16; -- O tamanho de cada palavra da memória  
ADDRESS\_RADIX = UNS; -- A base do endereço  
DATA\_RADIX = BIN; -- A base dos valores armazenados  
CONTENT -- Os valores a serem inseridos na memoria

### 2.2.4) Mostre o que deve ser modificado no código VHDL de ram\_16x4.vhd para ampliar a capacidade da memória. Exemplifique para uma memória 32x8.

Na declaração da entidade, devemos modificar as seguintes linhas:

endereco : in std\_logic\_vector(5 downto 0);  
dados\_entrada : in std\_logic\_vector(7 downto 0);

Na arquitetura, devemos fazer as seguintes alterações:

type arranjo\_memoria is array(0 to 31) of std\_logic\_vector(7  
downto 0);

## 3) Elaboração do circuito “circuito\_exp3\_ativ1”

### 3.1) Descrição VHDL do circuito

---

-- Arquivo : contador\_163.vhd  
-- Projeto : Experiencia 01 - Primeiro Contato com VHDL

---

-- Descricao : contador binario hexadecimal (modulo 16)  
-- similar ao CI 74163

---

-- Revisoes :  
-- Data Versao Autor Descricao  
-- 29/12/2020 1.0 Edson Midorikawa criacao

---

library IEEE;  
use IEEE.std\_logic\_1164.all;

```

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

```

```
end if;
Q <= std_logic_vector(to_unsigned(IQ, Q'length));
-- a saída Q recebe o valor do sinal utilizado para a contagem
end process; -- fim do process
end comportamental; -- fim da arquitetura
```

---

```
-- Arquivo : comparador_85.vhd
-- Projeto : Experiencia 02 - Um Fluxo de Dados Simples
```

---

```
-- Descricao : comparador binario de 4 bits
--             similar ao CI 7485
--             baseado em descricao criada por Edson Gomi (11/2017)
```

---

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

---

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

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

```

    o_ALTB : out std_logic;
    o_AEQB : out std_logic
);
end entity comparador_85;

architecture dataflow of comparador_85 is -- inicio da arquitetura do comparador
-- sinais intermediarios que comparam A e B sem levar em consideracao as
entradas de cascateamento
signal agtb : std_logic;
signal aeqb : std_logic;
signal altb : std_logic;
begin
-- equacoes dos sinais: pagina 462, capitulo 6 do livro-texto
-- Wakerly, J.F. Digital Design - Principles and Practice, 4th Edition
-- veja tambem datasheet do CI SN7485 (Function Table)
    agtb <= (i_A3 and not(i_B3)) or
              (not(i_A3 xor i_B3) and i_A2 and not(i_B2)) or
              (not(i_A3 xor i_B3) and not(i_A2 xor i_B2) and i_A1 and not(i_B1))
or
              (not(i_A3 xor i_B3) and not(i_A2 xor i_B2) and not(i_A1 xor i_B1)
and i_A0 and not(i_B0));
-- checa se a > b
    aeqb <= not((i_A3 xor i_B3) or (i_A2 xor i_B2) or (i_A1 xor i_B1) or (i_A0 xor
i_B0));
-- checa se a = b
    altb <= not(agtб 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 03 - Desenvolvendo o Fluxo de Dados

---

-- Descricao : módulo de memória RAM 16x4

```
-- sinais we e ce ativos em baixo
-- codigo ADAPTADO do código encontrado no livro
-- VHDL Descricao e Sintese de Circuitos Digitais
-- Roberto D'Amore, LTC Editora.
```

---

```
-- Revisoes :
```

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

---

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

```
entity ram_16x4 is -- declaracao da entidade
port ( -- entradas
    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) -- saida
);
end entity ram_16x4;
```

```
architecture ram1 of ram_16x4 is -- inicio da arquitetura da memoria
type arranjo_memoria is array(0 to 15) of std_logic_vector(3 downto 0);
-- array de 16 posicoes de vetores de 4 bits
signal memoria : arranjo_memoria;
attribute ram_init_file: string;
attribute ram_init_file of memoria: signal is "ram_conteudo_inicial.mif";
-- arquivo mif de inicializacao da memoria
begin
```

```
process(ce, we, endereco, memoria) -- process que define o funcionamento da
memoria
begin
    if ce = '0' then -- dado armazenado na subida de "we" com "ce=0"
        if rising_edge(we)
            then memoria(to_integer(unsigned(endereco))) <= dado_entrada; -- escrita de “dado_entrada” na posicao de memoria “endereco”
```

```

        end if;
    end if;
    dado_saida <= memoria(to_integer(unsigned(endereco))); -- leitura do
dado armazenado na posicao de memoria "endereco"
end process;

end architecture ram1; -- fim da arquitetura

```

```

library ieee;
use ieee.std_logic_1164.all;

entity circuito_exp3_ativ1 is
    port (
        clock : in std_logic;
        reset : in std_logic;
        enable : in std_logic;
        escreve: in std_logic;
        chaves : in std_logic_vector (3 downto 0);
        igual : out std_logic;
        fim: out std_logic;
        db_contagem : out std_logic_vector (3 downto 0);
        db_memoria: out std_logic_vector(3 downto 0)
    );
end entity circuito_exp3_ativ1;

```

```
architecture estrutural of circuito_exp3_ativ1 is
```

```

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

```

```
end component;
```

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

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

```
signal rco_out, great,enable_in, reset_baixo,igual_out,menor: std_logic;
signal contador_out, memoria_out : std_logic_vector (3 downto 0);
begin
  reset_baixo<=reset;
  contador: contador_163 port map (
```

```

clock => clock,
clr => not reset_baixo,
ld    => '1',
ent   => enable_in,
enp   =>enable_in,
D     => "0000",
Q     => contador_out,
rco   => rco_out
);
    enable_in<=enable;
    db_contagem <= contador_out;
comparador: comparador_85 port map (
i_A3 =>memoria_out(3),
i_B3 => chaves(3),
i_A2 =>memoria_out(2),
i_B2 => chaves(2),
i_A1 =>memoria_out(1),
i_B1 => chaves(1),
i_A0 =>memoria_out(0),
i_B0 => chaves(0),
i_AGTB =>'0',
i_ALTB => '0',
i_EQQB => '1',
-- saidas
o_AGTB =>great,
o_ALTB =>menor,
o_EQQB => igual_out
);

memoria: ram_16x4 port map(
endereco => contador_out,
dados_entrada => chaves,
we => escreve,
ce => '0',
dados_saida => memoria_out
);
    db_memoria <= memoria_out;
igual<=igual_out;
fim<= rco_out;
end architecture;

```

### 3.2) Diagrama gerado através da ferramenta RTL Viewer

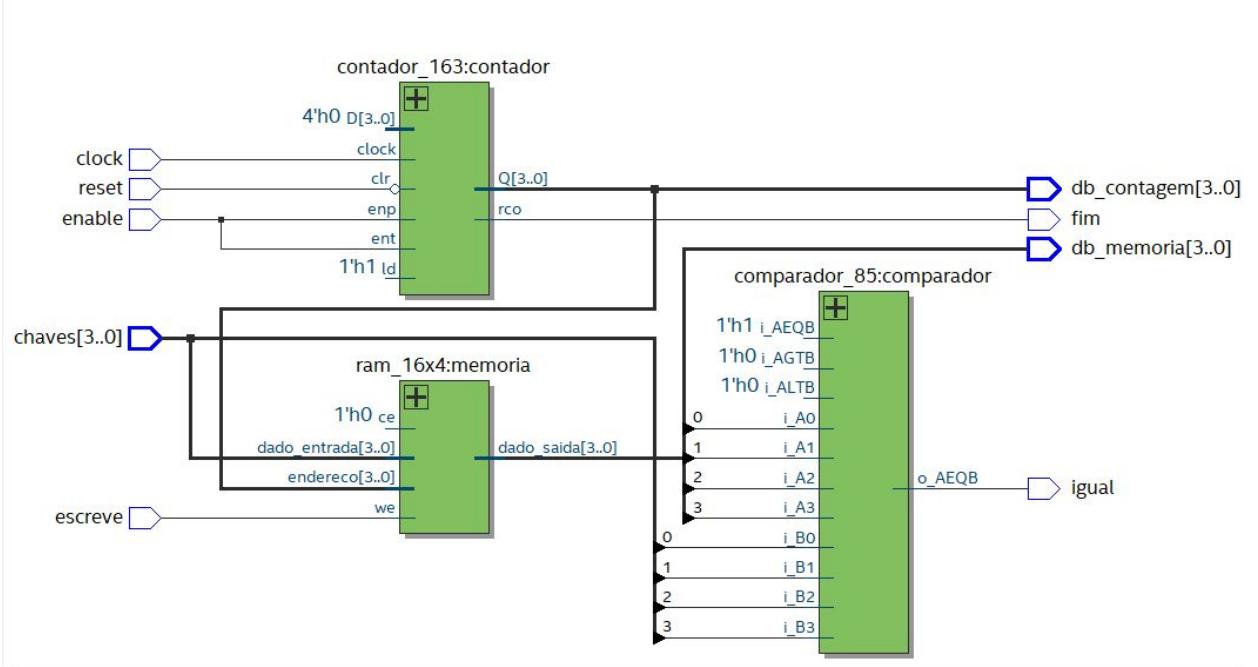


Diagrama gerado

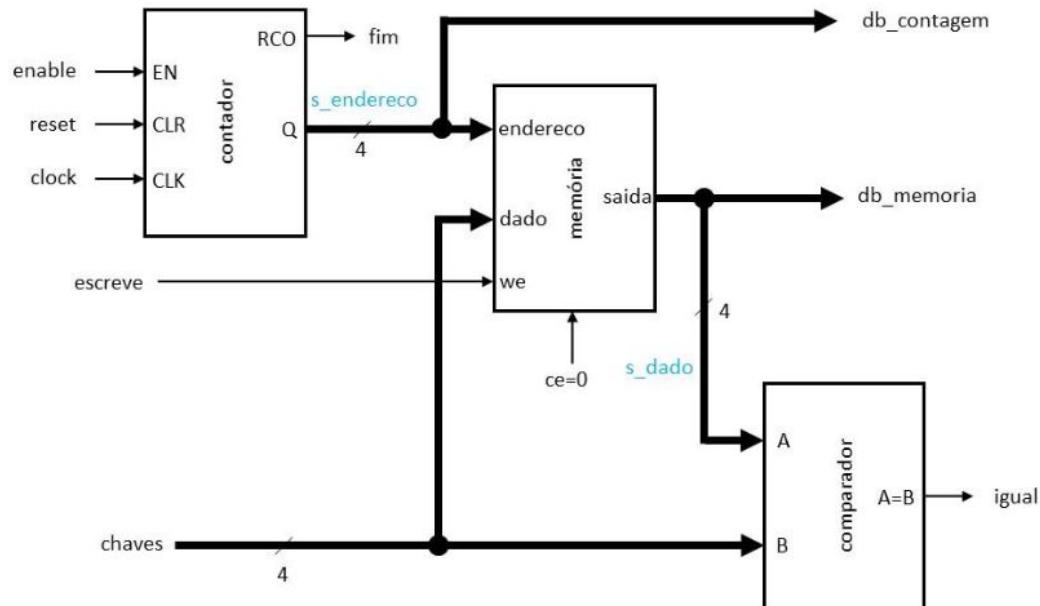


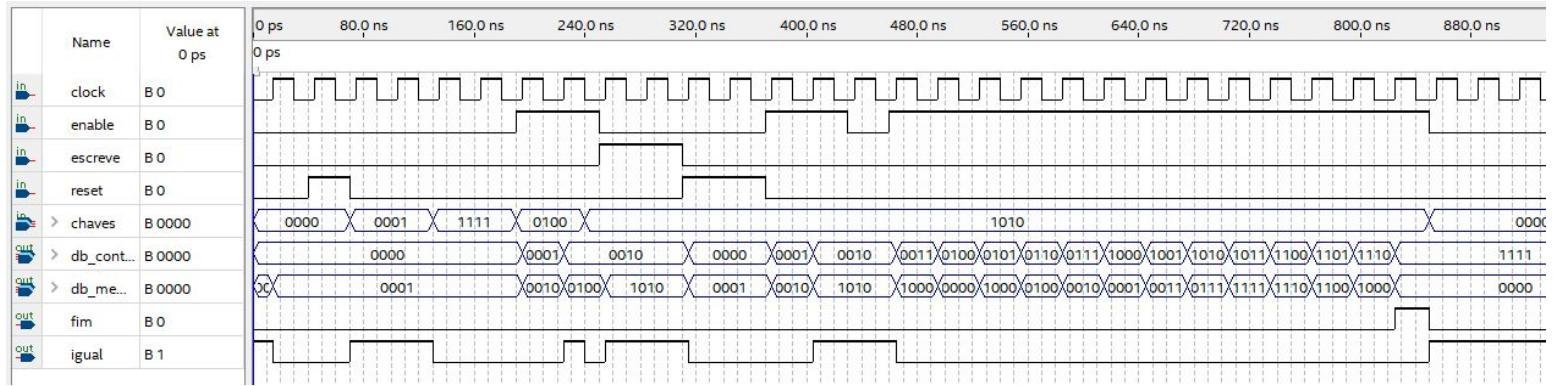
Diagrama fornecido na apostila da experiência

Comparando os diagramas, podemos ver que o circuito proposto foi devidamente implementado.

### 3.3) Plano de testes

#	Operação	Sinais de controle	Resultado esperado	Resultado observado
c.i.	Condições iniciais	clock=0 enable=0 reset=0 escreve=0 chaves=0000	igual=0 fim=0 db_contagem=0000 db_memoria=0001	igual=0 fim=0 db_contagem=0000 db_memoria=0001
1	Zerar contador e observar a saída da memória	reset=1 clock	db_contagem=0000 db_memoria=0001	db_contagem=0000 db_memoria=0001
2	Ajustar chaves para 0001	chaves=0001	igual=1	igual=1
3	Ajustar chaves para 1111	chaves=1111	igual=0	igual=0
4	Incrementar endereço para 2 e ajustar chaves para 0100	enable=1 Aguardar 2 pulsos de clock chaves=0100	db_memoria=0100 igual=1	db_memoria=0100 igual=1
5	Ajustar chaves para 1010 e acionar sinal “escreve” (escrita do dado 1010 no endereço 2 da memória)	enable=0 chaves=1010 escreve=1	db_memoria=1010	db_memoria=1010
6	Zerar contador	reset=1 clock	db_contagem=0000 db_memoria=0001	db_contagem=0000 db_memoria=0001
7	Incrementar endereço para 2 (verificar dado gravado)	enable=1 Aguardar 2 pulsos de clock	db_contagem=0010 db_memoria=1010	db_contagem=0010 db_memoria=1010
8	Incrementar endereço até F	Aguardar 12 pulsos de clock	db_contagem=1111 db_memoria=0000 fim=1	db_contagem=1111 db_memoria=0000 fim=1
9	Ajustar chaves para 0000	chaves=0000	igual=1	igual=1

### 3.4) Simulação do funcionamento do circuito



### 3.5) Designação de pinos para o circuito

Sinal	Pino na Placa DE0-CV	Pino no FPGA	Analog Discovery
CLOCK	GPIO_0_D13	PIN_T22	StaticIO – Button 0/1 – DIO0
RESET	GPIO_0_D15	PIN_N19	StaticIO – Switch Push/Pull – DIO1
ENABLE	GPIO_0_D17	PIN_P19	StaticIO – Switch Push/Pull – DIO2
ESCREVE	GPIO_0_D19	PIN_P17	StaticIO – Switch Push/Pull – DIO3
CHAVE(0)	GPIO_0_D21	PIN_M18	StaticIO – Switch Push/Pull – DIO4
CHAVE(1)	GPIO_0_D23	PIN_L17	StaticIO – Switch Push/Pull – DIO5
CHAVE(2)	GPIO_0_D25	PIN_K17	StaticIO – Switch Push/Pull – DIO6
CHAVE(3)	GPIO_0_D27	PIN_P18	StaticIO – Switch Push/Pull – DIO7
DB_CONTAGEM(0)	Led LEDR0	PIN_AA2	-
DB_CONTAGEM(1)	Led LEDR1	PIN_AA1	-
DB_CONTAGEM(2)	Led LEDR2	PIN_W2	-
DB_CONTAGEM(3)	Led LEDR3	PIN_Y3	-
FIM	Led LEDR4	PIN_N2	-
DB_MEMORIA(0)	Led LEDR5	PIN_N1	-
DB_MEMORIA(1)	Led LEDR6	PIN_U2	-
DB_MEMORIA(2)	Led LEDR7	PIN_U1	-
DB_MEMORIA(3)	Led LEDR8	PIN_L2	-
IGUAL	Led LEDR9	PIN_L1	-

#### 4) Elaboração do circuito “circuito\_exp3\_ativ2”

##### 4.1) Descrição VHDL do circuito

---

-- Arquivo : contador\_163.vhd

-- Projeto : Experiencia 01 - Primeiro Contato com VHDL

---

-- Descricao : contador binario hexadecimal (modulo 16)

-- similar ao CI 74163

---

-- Revisoes :

-- Data Versao Autor Descricao

-- 29/12/2020 1.0 Edson Midorikawa criacao

---

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity contador_163 is -- entidade principal
port (
    clock : in std_logic; -- sinais de entrada
    clr : in std_logic;
    ld : in std_logic;
    ent : in std_logic;
    enp : in std_logic;
    D : in std_logic_vector (3 downto 0);
    Q : out std_logic_vector (3 downto 0); -- sinais de saída
    rco : out std_logic
);
end contador_163;
architecture comportamental of contador_163 is -- declaração da
arquitetura
signal IQ: integer range 0 to 15;
begin
process (clock,ent,IQ) -- inicio do process do circuito
begin
if clock'event and clock='1' then
-- as mudanças no circuito ocorrem com o clock em 1
if clr='0' then IQ <= 0;
-- caso o sinal clear seja 0, a contagem é reiniciada
elsif ld='0' then IQ <= to_integer(unsigned(D));
```

```

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

```

---

-- Arquivo : comparador\_85.vhd

-- Projeto : Experiencia 02 - Um Fluxo de Dados Simples

---

-- Descricao : comparador binario de 4 bits

-- similar ao CI 7485

-- baseado em descricao criada por Edson Gomi (11/2017)

---

-- Revisoes :

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

---

library ieee;

```

use ieee.std_logic_1164.all;

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

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

```

```

aeqb <= not((i_A3 xor i_B3) or (i_A2 xor i_B2) or (i_A1 xor i_B1) or (i_A0
xor i_B0));
-- checa se a = b
alrb <= not(agt or aeqb);
-- checa se a < b
o_AGTB <= agt or (aeqb and (not(i_EQB) and not(i_ALTB)));
o_ALTB <= alrb or (aeqb and (not(i_EQB) and not(i_AGTB)));
o_EQB <= aeqb and i_EQB;
-- 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 03 - Desenvolvendo o Fluxo de Dados

---

-- Descricao : módulo de memória RAM 16x4  
--        sinais we e ce ativos em baixo  
--        codigo ADAPTADO do código encontrado no livro  
--        VHDL Descricao e Sintese de Circuitos Digitais  
--        Roberto D'Amore, LTC Editora.

---

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

---

```

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

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

```

```

end entity ram_16x4;

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

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

end architecture ram1;

```

---

-- Arquivo : hexa7seg.vhd

-- Projeto : Jogo do Desafio da Memoria

---

-- Descricao : decodificador hexadecimal para  
-- display de 7 segmentos

--  
-- entrada: hexa - codigo binario de 4 bits hexadecimal  
-- saida: sseg - codigo de 7 bits para display de 7 segmentos

---

-- dica de uso: mapeamento para displays da placa DE0-CV  
-- bit 6 mais significativo é o bit a esquerda  
-- p.ex. sseg(6) -> HEX0[6] ou HEX06

---

-- Revisoes :

-- Data	Versao	Autor	Descricao
-- 29/12/2020	1.0	Edson Midorikawa	criacao

```
-----  
library ieee;  
use ieee.std_logic_1164.all;  
  
entity hexa7seg is  
    port (  
        hexa : in std_logic_vector(3 downto 0);  
        sseg : out std_logic_vector(6 downto 0)  
    );  
end hexa7seg;  
  
architecture comportamental of hexa7seg is  
begin  
  
    sseg <= "1000000" when hexa="0000" else  
    "1111001" when hexa="0001" else  
    "0100100" when hexa="0010" else  
    "0110000" when hexa="0011" else  
    "0011001" when hexa="0100" else  
    "0010010" when hexa="0101" else  
    "0000010" when hexa="0110" else  
    "1111000" when hexa="0111" else  
    "0000000" when hexa="1000" else  
    "0000000" when hexa="1000" else  
    "0010000" when hexa="1001" else  
    "0001000" when hexa="1010" else  
    "0000011" when hexa="1011" else  
    "1000110" when hexa="1100" else  
    "0100001" when hexa="1101" else  
    "0000110" when hexa="1110" else  
    "0001110" when hexa="1111" else  
    "1111111";  
  
end comportamental;
```

```

library ieee;
use ieee.std_logic_1164.all;

entity circuito_exp3_ativ2 is
    port (
        clock : in std_logic;
        reset : in std_logic;
        enable : in std_logic;
        escreve: in std_logic;
        chaves : in std_logic_vector (3 downto 0);
        igual : out std_logic;
        fim: out std_logic;
        db_contagem : out std_logic_vector (6 downto 0);
        db_memoria: out std_logic_vector(6 downto 0)
    );
end entity circuito_exp3_ativ2;

architecture estrutural of circuito_exp3_ativ2 is

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

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

```

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

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

```

signal rco_out, great,enable_in, reset_baixo,igual_out,menor: std_logic;
signal contador_out, memoria_out : std_logic_vector (3 downto 0);
begin
    reset_baixo<= not reset;
    contador: contador_163 port map (
        clock => clock,
        clr => reset_baixo,
        ld  => '1',
        ent => enable_in,
        enp =>enable_in,
        D   => "0000",
        Q   => contador_out,
        rco => rco_out
    );
    enable_in<=enable;

seg_conta: hexa7seg port map(
    hexa => contador_out,
    sseg=>db_contagem
);
comparador: comparador_85 port map (
    i_A3 =>memoria_out(3),
    i_B3  => chaves(3),
    i_A2  =>memoria_out(2),
    i_B2  => chaves(2),
    i_A1  =>memoria_out(1),
    i_B1  => chaves(1),
    i_A0  =>memoria_out(0),
    i_B0  => chaves(0),
    i_AGTB =>'0',
    i_ALTB => '0',
    i_AEQB => '1',
    -- saidas
    o_AGTB =>great,
    o_ALTB =>menor,
    o_AEQB => igual_out
);
memoria: ram_16x4 port map(
    endereco => contador_out,

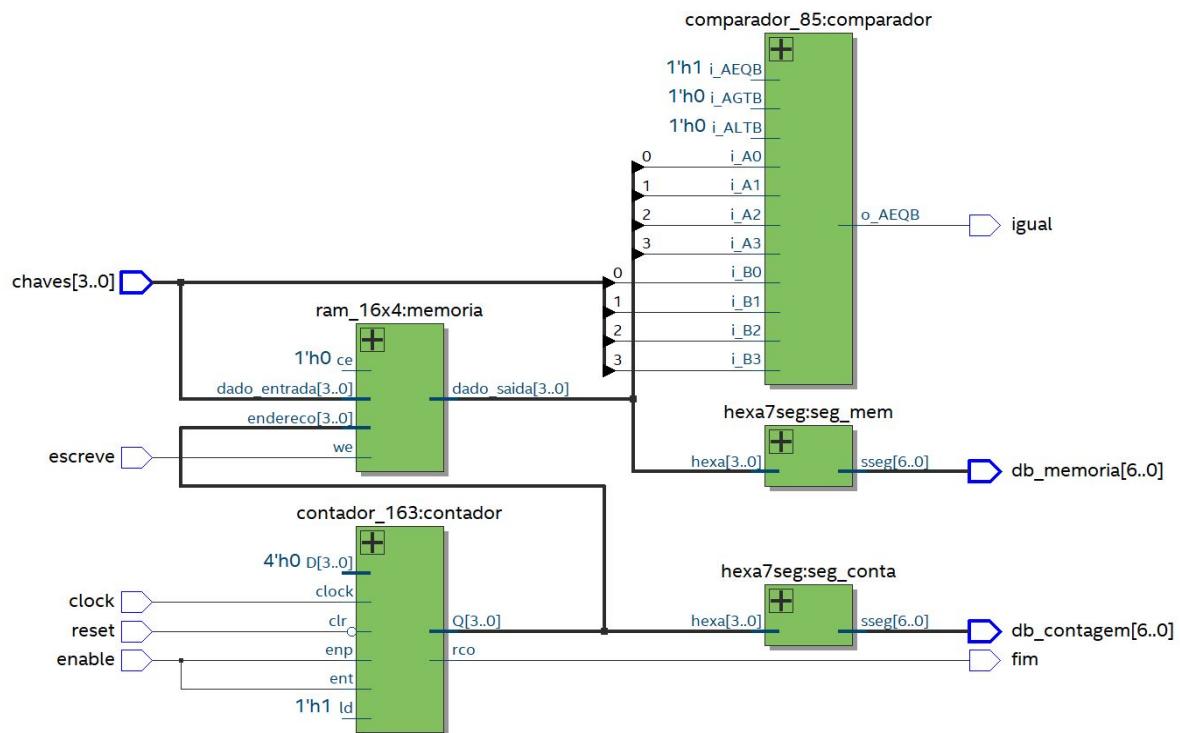
```

```

    dado_entrada => chaves,
    we => escreve,
    ce => '0',
    dado_saida => memoria_out
);
seg_mem: hexa7seg port map(
    hexa => memoria_out,
    sseg=>db_memoria
);
    igual<=igual_out;
fim<= rco_out;
end architecture;

```

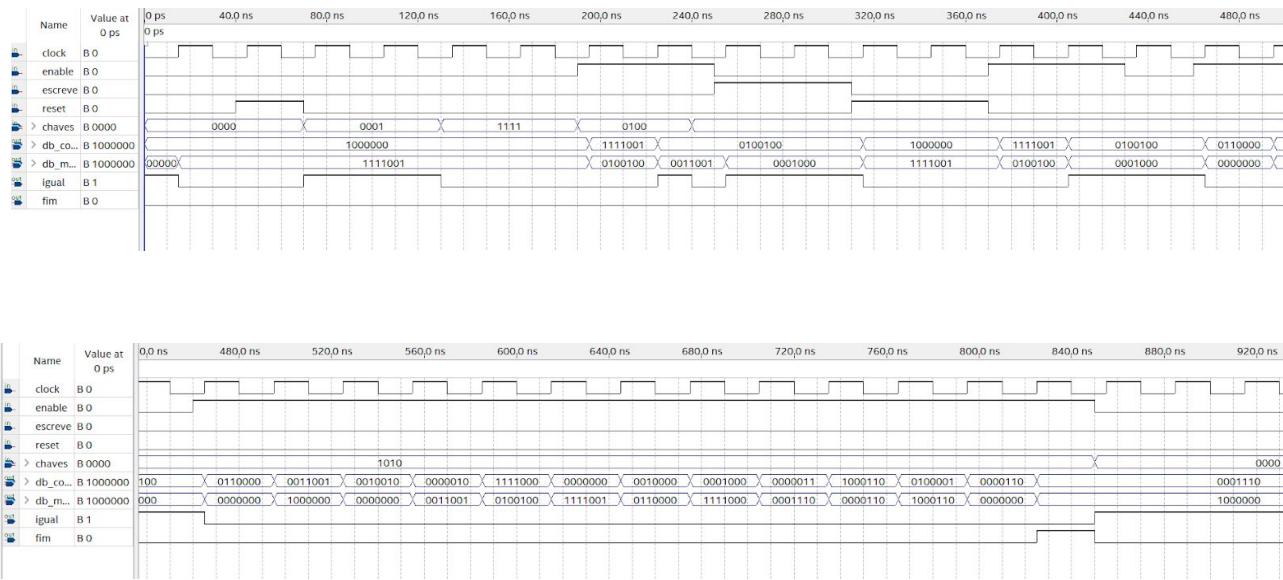
#### 4.2) Diagrama gerado através da ferramenta RTL Viewer



### 4.3) Plano de testes

#	Operação	Sinais de controle	Resultado esperado	Resultado observado
c.i.	Condições iniciais	clock=0 enable=0 reset=0 escreve=0 chaves=0000	igual=0 fim=0 db_contagem=1000000 db_memoria=1111001	igual=0 fim=0 db_contagem=1000000 db_memoria=1111001
1	Zerar contador e observar a saída da memória	reset=1 clock	db_contagem=1000000 db_memoria=1111001	db_contagem=1000000 db_memoria=1111001
2	Ajustar chaves para 0001	chaves=0001	igual=1	igual=1
3	Ajustar chaves para 1111	chaves=1111	igual=0	igual=0
4	Incrementar endereço para 2 e ajustar chaves para 0100	enable=1 Aguardar 2 pulsos de clock chaves=0100	db_memoria=0011001 igual=1	db_memoria=0011001 igual=1
5	Ajustar chaves para 1010 e acionar sinal “escreve” (escrita do dado 1010 no endereço 2 da memória)	enable=0 chaves=1010 escreve=1	db_memoria=0001000	db_memoria=0001000
6	Zerar contador	reset=1 clock	db_contagem=1000000 db_memoria=1111001	db_contagem=1000000 db_memoria=1111001
7	Incrementar endereço para 2 (verificar dado gravado)	enable=1 Aguardar 2 pulsos de clock	db_contagem=0100100 db_memoria=0001000	db_contagem=0100100 db_memoria=0001000
8	Incrementar endereço até F	Aguardar 12 pulsos de clock	db_contagem=0001110 db_memoria=1000000 fim=1	db_contagem=0001110 db_memoria=1000000 fim=1
9	Ajustar chaves para 0000	chaves=0000	igual=1	igual=1

#### 4.4) Simulação do funcionamento do circuito



## 4.5) Designação de pinos para o circuito

Sinal	Pino na Placa DE0-CV	Pino no FPGA	Analog Discovery
CLOCK	GPIO_0_D13	PIN_T22	StaticIO – Button 0/1 – DIO0
RESET	GPIO_0_D15	PIN_N19	StaticIO – Switch Push/Pull – DIO1
ENABLE	GPIO_0_D17	PIN_P19	StaticIO – Switch Push/Pull – DIO2
ESCREVE	GPIO_0_D19	PIN_P17	StaticIO – Switch Push/Pull – DIO3
CHAVE(0)	GPIO_0_D21	PIN_M18	StaticIO – Switch Push/Pull – DIO4
CHAVE(1)	GPIO_0_D23	PIN_L17	StaticIO – Switch Push/Pull – DIO5
CHAVE(2)	GPIO_0_D25	PIN_K17	StaticIO – Switch Push/Pull – DIO6
CHAVE(3)	GPIO_0_D27	PIN_P18	StaticIO – Switch Push/Pull – DIO7
DB_CONTAGEM(0)	HEX00	PIN_U21	-
DB_CONTAGEM(1)	HEX01	PIN_V21	-
DB_CONTAGEM(2)	HEX02	PIN_W22	-
DB_CONTAGEM(3)	HEX03	PIN_W21	
DB_CONTAGEM(4)	HEX04	PIN_Y22	
DB_CONTAGEM(5)	HEX05	PIN_Y21	
DB_CONTAGEM(6)	HEX06	PIN_AA22	-
FIM	Led LEDR4	PIN_N2	-
DB_MEMORIA(0)	HEX10	PIN_AA20	-
DB_MEMORIA(1)	HEX11	PIN_AB20	-
DB_MEMORIA(2)	HEX12	PIN_AA19	-
DB_MEMORIA(3)	HEX13	PIN_AA18	
DB_MEMORIA(4)	HEX14	PIN_AB18	
DB_MEMORIA(5)	HEX15	PIN_AA17	
DB_MEMORIA(6)	HEX16	PIN_U22	-
IGUAL	Led LEDR9	PIN_L1	-

## 5) Desafio

### 5.1) Descrição VHDL do Circuito

---

-- Arquivo : contador\_163.vhd  
-- Projeto : Experiencia 01 - Primeiro Contato com VHDL

---

-- Descricao : contador binario hexadecimal (modulo 16)  
-- similar ao CI 74163

---

-- Revisoes :

-- Data Versao Autor Descricao

-- 29/12/2020 1.0 Edson Midorikawa criacao

---

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

```

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

```

---

-- Arquivo : comparador\_85.vhd  
 -- Projeto : Experiencia 02 - Um Fluxo de Dados Simples

---

-- Descricao : comparador binario de 4 bits  
 -- similar ao CI 7485  
 -- baseado em descricao criada por Edson Gomi (11/2017)

---

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

---

```

library ieee;
use ieee.std_logic_1164.all;

entity comparador_85 is -- declaracao da entidade do comparador
port ( -- entradas

```

```

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

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

```

```

o_AGTB <= agtb or (aeqb and (not(i_AEQB) and not(i_ALTB)));
o_ALTB <= altb or (aeqb and (not(i_AEQB) and not(i_AGTB)));
o_AEQB <= aeqb and i_AEQB;
    -- nas saidas, 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 03 - Desenvolvendo o Fluxo de Dados

---

-- Descricao : módulo de memória RAM 16x4  
-- sinais we e ce ativos em baixo  
-- código ADAPTADO do código encontrado no livro  
-- VHDL Descricao e Sintese de Circuitos Digitais  
-- Roberto D'Amore, LTC Editora.

---

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

---

```

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

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

```

```

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

```

```

attribute ram_init_file: string;
attribute ram_init_file of memoria: signal is "ram_conteudo_inicial.mif";
begin

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

end architecture ram1;

```

---

-- Arquivo : hexa7seg.vhd

-- Projeto : Jogo do Desafio da Memoria

---

-- Descricao : decodificador hexadecimal para

-- display de 7 segmentos

--

-- entrada: hexa - codigo binario de 4 bits hexadecimal

-- saida: sseg - codigo de 7 bits para display de 7 segmentos

---

-- dica de uso: mapeamento para displays da placa DE0-CV

-- bit 6 mais significativo é o bit a esquerda

-- p.ex. sseg(6) -> HEX0[6] ou HEX06

---

-- Revisoes :

-- Data      Versao    Autor      Descricao

-- 29/12/2020 1.0   Edson Midorikawa criacao

---

```

library ieee;
use ieee.std_logic_1164.all;

```

```

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

architecture comportamental of hexa7seg is
begin

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

```

```

library ieee;
use ieee.std_logic_1164.all;

```

```

entity circuito_exp3_desafio is
  port (
    clock : in std_logic;
    zeraL : in std_logic;
    zeraC : in std_logic;
    enableL : in std_logic;
    enableC : in std_logic;
    escreve: in std_logic;
    chaves : in std_logic_vector (3 downto 0);
    fimL: out std_logic;
    fimC: out std_logic;
    db_contagem : out std_logic_vector (6 downto 0);
    db_memoria: out std_logic_vector(6 downto 0);
    db_limite: out std_logic_vector(6 downto 0);
    enderecoMenorLimite: out std_logic;
    chavesIgualMemoria: out std_logic
  );
end entity circuito_exp3_desafio;

```

```

architecture estrutural of circuito_exp3_desafio is

```

```

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

```

```

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

```

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

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

```

      signal      rco_out,      great,enable_lin,
ZeraL_baixo,ZeraC_baixo,igual_out,menor,great_o,menor_o,enable_cin,igu
al_out1,rco_outl,rco_outc: std_logic;
      signal contador_out, s_dado ,s_limite,s_endereco: std_logic_vector (3
downto 0);
begin
      ZeraL_baixo<= not ZeraL;
ContLmt: contador_163 port map (
      clock => clock,
      clr => ZeraL_baixo,
      Id  => '1',
      ent  => enable_Lin,
      enp  =>enable_Lin,
      D   => "0000",
      Q   => s_limite,
      rco  => rco_outL
);
      enable_Lin<=enableL;

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

seg_contlim: hexa7seg port map(
      hexa => s_limite,
      sseg=>db_limite
);

seg_contend: hexa7seg port map(
      hexa => s_endereco,
      sseg=>db_contagem

```

```

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

compEnd: comparador_85 port map (
    i_A3 =>s_limite(3),
    i_B3 => s_endereco(3),
    i_A2 =>s_limite(2),
    i_B2 => s_endereco(2),
    i_A1 =>s_limite(1),
    i_B1 => s_endereco(1),
    i_A0 =>s_limite(0),
    i_B0 => s_endereco(0),
    i_AGTB =>'0',
    i_ALTB => '1',
    i_AEQB => '0',
    -- saidas
    o_AGTB =>great_o,
    o_ALTB =>menor_o,
    o_AEQB => igual_out1
);

memoria: ram_16x4 port map(
    endereco => s_endereco,
    dado_entrada => chaves,

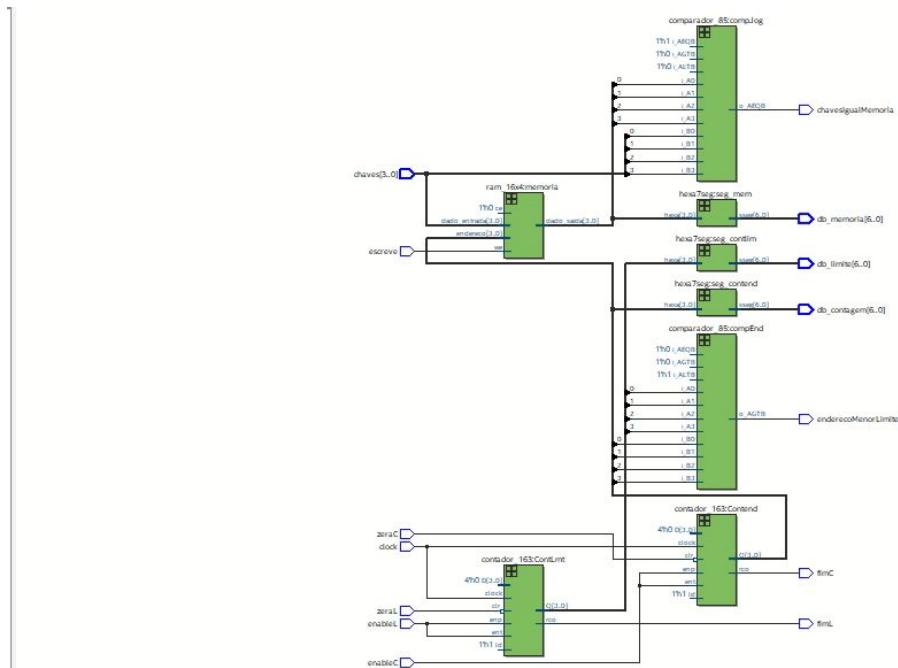
```

```

we => escreve,
ce => '0',
dado_saida => s_dado
);
seg_mem: hexa7seg port map(
    hexa => s_dado,
    sseg=>db_memoria
);
chavesIgualMemoria<=igual_out;
enderecoMenorLimite<=great_o;
fimL<= rco_outL;
fimC<=rco_outC;
end architecture;

```

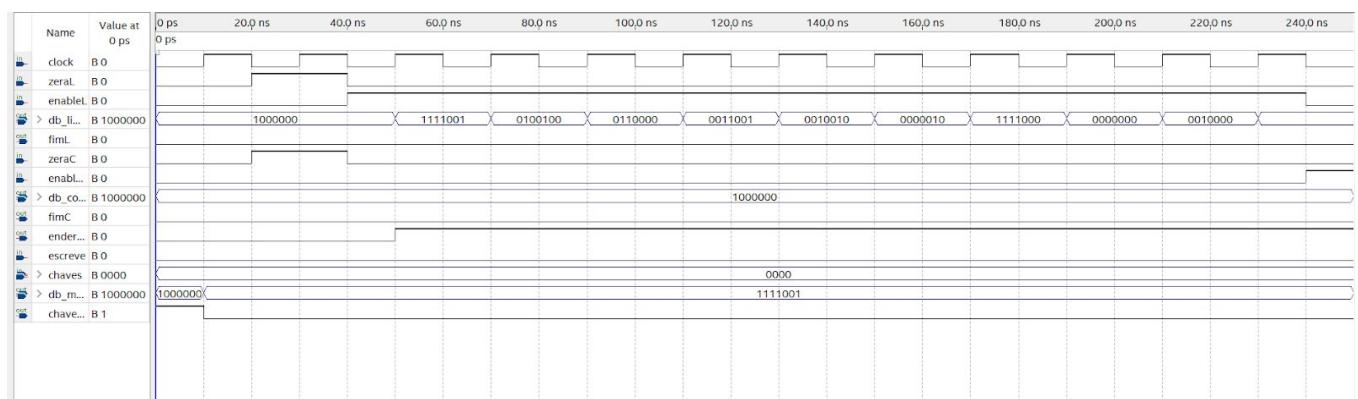
## 5.2) Diagrama gerado através da ferramenta RTL Viewer

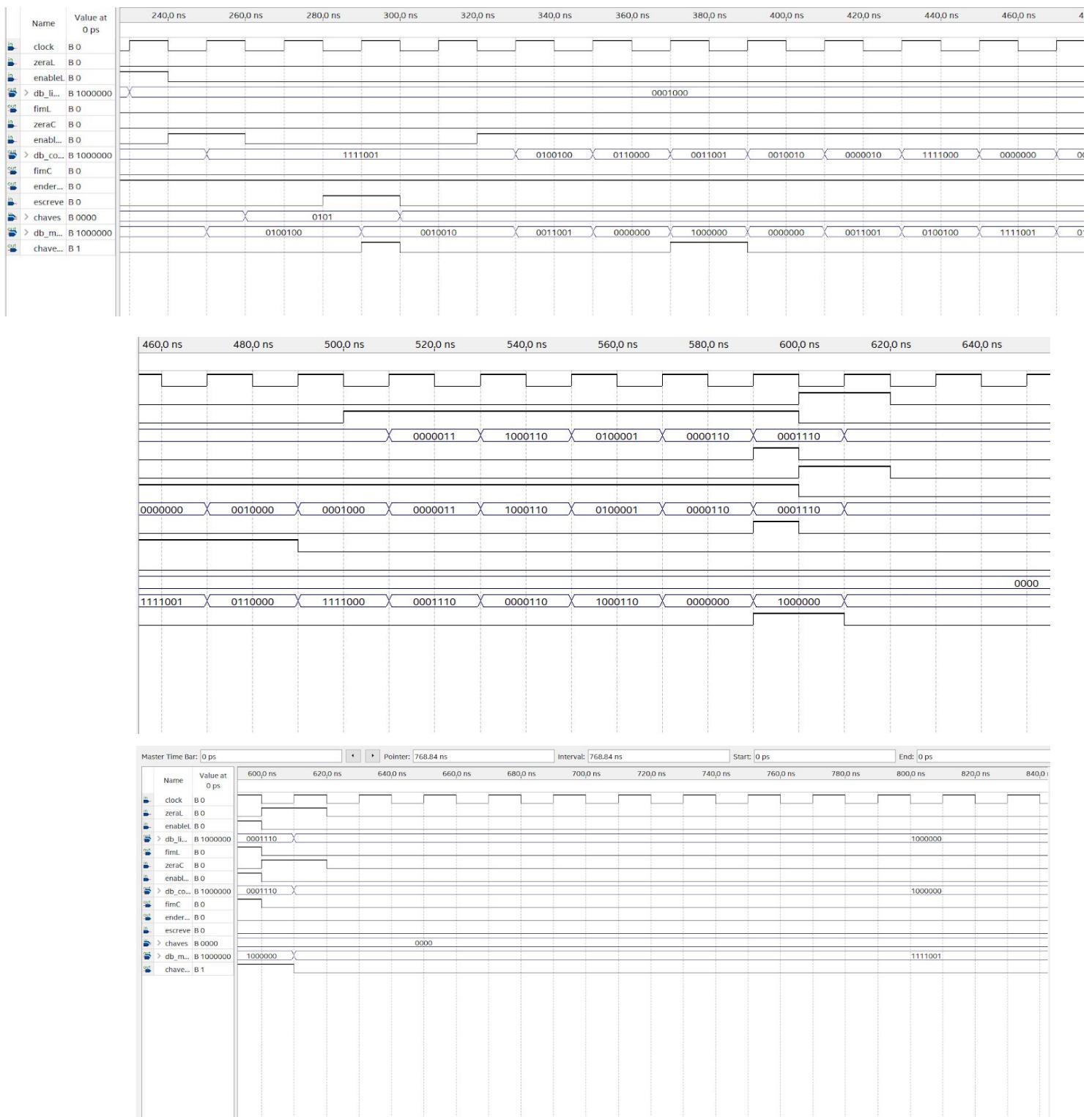


### 5.3) Plano de testes

#	Operação	Sinais de controle	Resultado esperado	Resultado observado
c.i.	Condições iniciais	clock=0 zeral=0, enableL=0, zeraC=0, enableC=0, escreve=0, chaves=0000	db_limite= 1000000, db_contagem=1000000, fimL=0, fimC=0, db_memoria=1111001, endereço menor que limite=0, chavesigualmemoria=1 —	db_limite= 1000000, db_contagem=1000000, fimL=0, fimC=0, db_memoria=1111001, endereço menor que limite=0, chavesigualmemoria=1
1	Zerar contadores	zeraC=1, zeraL=1 aguardar 1 clock	db_limite= 1000000, db_contagem=1000000	db_limite= 1000000, db_contagem=1000000
2	Habilitar contador limite e contar até 10	zeral=0, zeraC=0, enableL=1, aguardar 10 clock	db_limite=0001000 endereço menor que limite=1,	db_limite=0001000 endereço menor que limite=1,
3	Desabilitar contador limite e habilitar contador endereços, contar até 1	EnableL=0, EnableC=1 Aguardar 1 clock	db_contagem = 1111001 db_memoria=0100100	db_contagem = 1111001 db_memoria=0100100
4	Desabilitar contador de endereços, chaves em 0101	enableC=0, chaves=0101 Aguardar um período de clock	db_contagem = 1111001 db_memoria=0100100	db_contagem = 1111001 db_memoria=0100100
5	Habilitar contador de endereços contar até 10	enableC=1 Aguardar 9 períodos de clock	db_contagem=0001000 endereço menor que limite=0	db_contagem=0001000 endereço menor que limite=0
6	Habilitar contador limite e contar até o fim	enable L=1 Aguardar 5 períodos de clock	db_contagem =0001110 db_limite=0001110 fimC=1 fimL=1	db_contagem =0001110 db_limite=0001110 fimC=1 fimL=1
7	Zerar contadores	ZeraL=1 zeraC=1 Aguardar um período de clock	db_limite= 1000000, db_contagem=1000000 fimC=0 fimL=0	db_limite= 1000000, db_contagem=1000000 fimC=0 fimL=0

### 5.4) Simulação do funcionamento do circuito





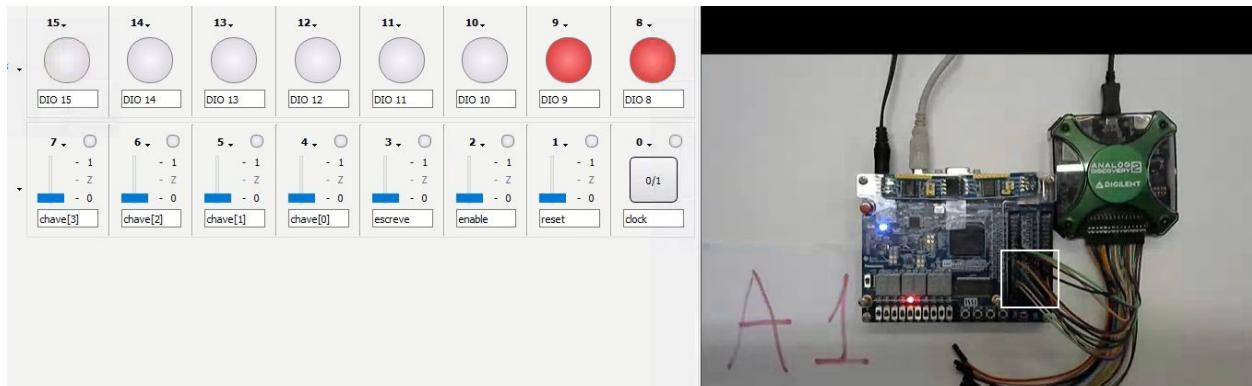
## 5.5) Designação de pinos para o circuito

Sinal	Pino na Placa DE0-CV	Pino na FPGA	Analog Discovery
CLOCK	GPIO_0_D13	PIN_T22	StaticIO – Button 0/1 – DIO0
ZERAC	GPIO_0_D15	PIN_N19	StaticIO – Switch Push/Pull – DIO1
ENABLEC	GPIO_0_D17	PIN_P19	StaticIO – Switch Push/Pull – DIO2
ESCREVE	GPIO_0_D19	PIN_P17	StaticIO – Switch Push/Pull – DIO3
CHAVE(0)	GPIO_0_D21	PIN_M18	StaticIO – Switch Push/Pull – DIO4
CHAVE(1)	GPIO_0_D23	PIN_L17	StaticIO – Switch Push/Pull – DIO5
CHAVE(2)	GPIO_0_D25	PIN_K17	StaticIO – Switch Push/Pull – DIO6
CHAVE(3)	GPIO_0_D27	PIN_P18	StaticIO – Switch Push/Pull – DIO7
ZERAL	GPIO_0_D29	PIN_R17	StaticIO – Switch Push/Pull – DIO8
ENABLEL	GPIO_0_D31	PIN_T20	StaticIO – Switch Push/Pull – DIO9
DB_CONTAGEM(0)	HEX00	PIN_U21	-
DB_CONTAGEM(1)	HEX01	PIN_V21	
DB_CONTAGEM(2)	HEX02	PIN_W22	
DB_CONTAGEM(3)	HEX03	PIN_W21	
DB_CONTAGEM(4)	HEX04	PIN_Y22	
DB_CONTAGEM(5)	HEX05	PIN_Y21	
DB_CONTAGEM(6)	HEX06	PIN_AA22	
DB_MEMORIA(0)	HEX10	PIN_AA20	-
DB_MEMORIA(1)	HEX11	PIN_AB20	
DB_MEMORIA(2)	HEX12	PIN_AA19	
DB_MEMORIA(3)	HEX13	PIN_AA18	
DB_MEMORIA(4)	HEX14	PIN_AB18	
DB_MEMORIA(5)	HEX15	PIN_AA17	
DB_MEMORIA(6)	HEX16	PIN_U22	

<b>DB_LIMITE(0)</b>	<b>HEX20</b>	<b>PIN_Y19</b>	-
<b>DB_LIMITE(1)</b>	<b>HEX21</b>	<b>PIN_AB17</b>	
<b>DB_LIMITE(2)</b>	<b>HEX22</b>	<b>PIN_AA10</b>	
<b>DB_LIMITE(3)</b>	<b>HEX23</b>	<b>PIN_Y14</b>	
<b>DB_LIMITE(4)</b>	<b>HEX24</b>	<b>PIN_V14</b>	
<b>DB_LIMITE(5)</b>	<b>HEX25</b>	<b>PIN_AB22</b>	
<b>DB_LIMITE(6)</b>	<b>HEX26</b>	<b>PIN_AB21</b>	
<b>FIMC</b>	<b>Led LEDR4</b>	<b>PIN_N2</b>	-
<b>FIML</b>	<b>Led LEDR5</b>	<b>PIN_N1</b>	-
<b>ENDERECOMENORLIMITE</b>	<b>Led LEDR8</b>	<b>PIN_L2</b>	-
<b>CHAVESIGUALMEMORIA</b>	<b>Led LEDR9</b>	<b>PIN_L1</b>	-

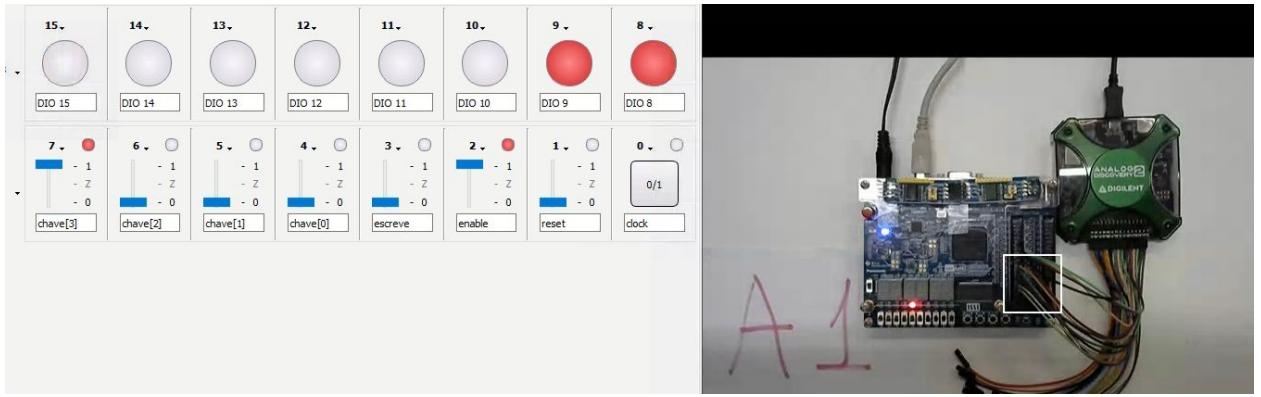
## 6) Demonstração do Funcionamento dos Circuitos

### 6.1) Circuito “circuito\_exp3\_ativ1”

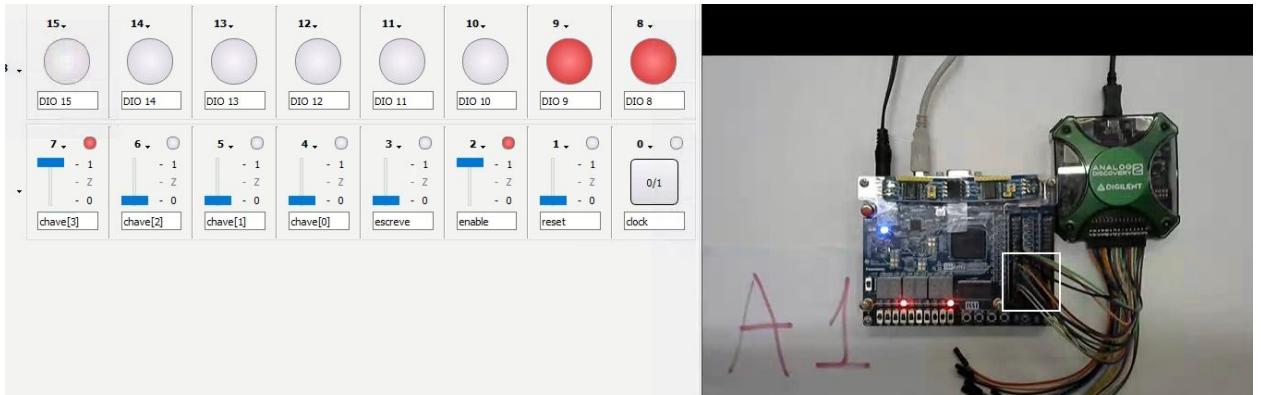


Condições iniciais

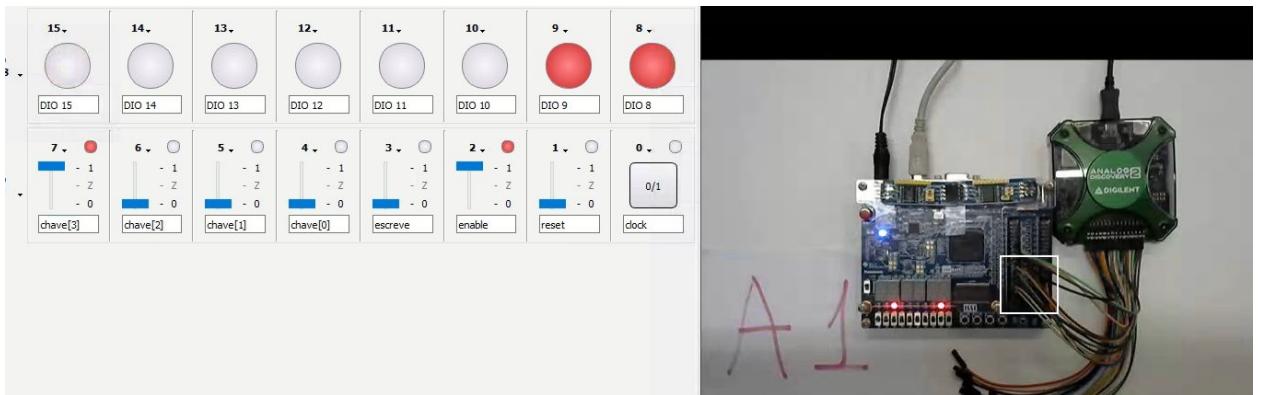
Da esquerda para a direita, os 4 primeiros LEDs indicam o valor do contador, o 5º LED nos mostra se a contagem chegou ao fim e os próximos 4 LEDs indicam o valor armazenado na posição de memória de mesmo valor do contador. Por fim, o último LED mostra se o valor lido na memória é igual ao valor de “chave”. Neste caso, o valor armazenado na posição 0000 é 0001, como definido no arquivo MIF que inicializou a memória.



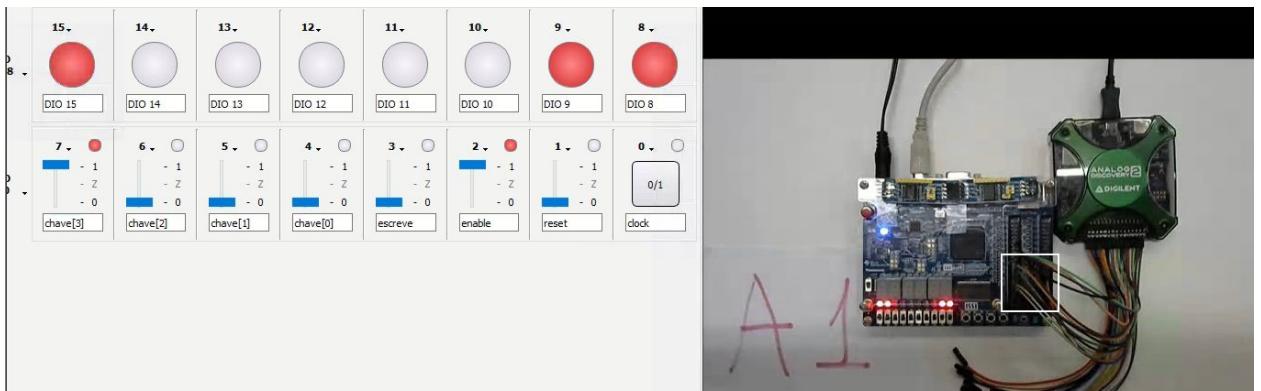
O valor de “enable” foi setado para 1 e o valor de “chave” para 1000.



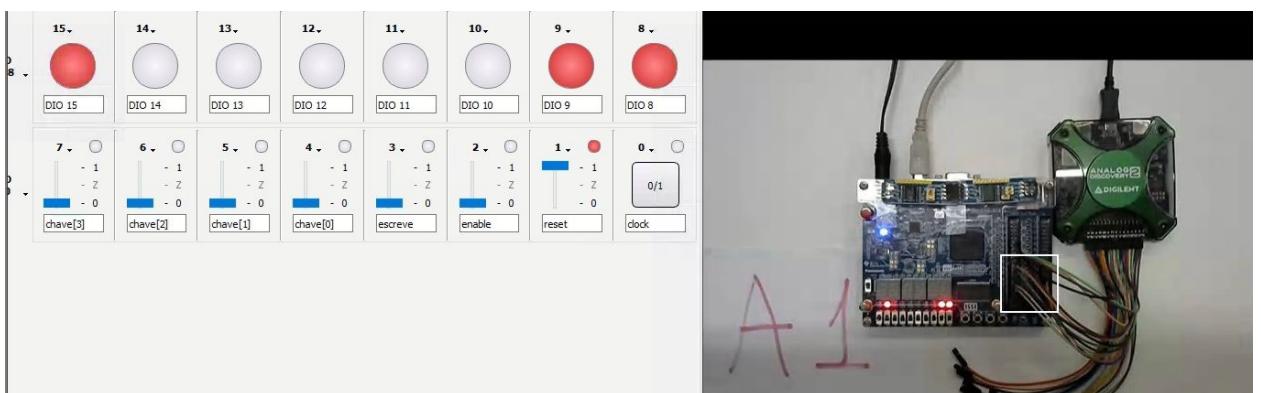
Após 1 pulso de clock, o valor do contador sobe para 0001 e o valor lido na memória é 0010.



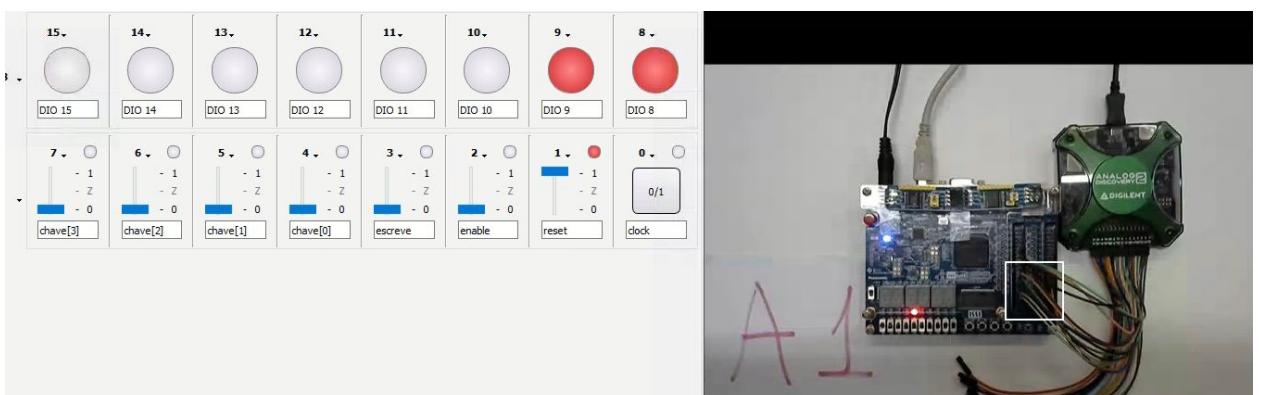
Após 1 pulso de clock, o valor do contador sobe para 0010 e o valor lido na memória é 0100.



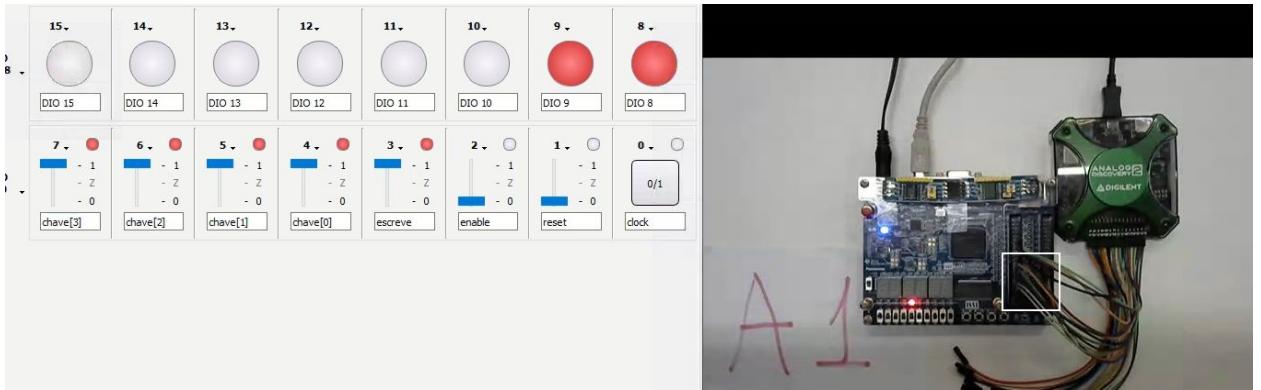
Após mais um pulso de clock, o valor do contador sobe para 0011 e o valor lido na memória é 1000, igual ao de “chaves” e, portanto, o LED ligado ao sinal “igual” acende.



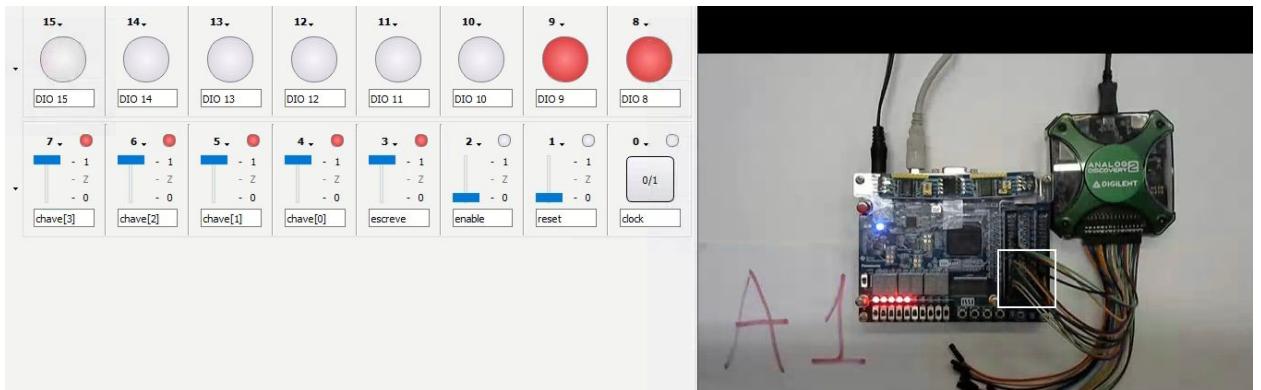
O valor de “chave” é setado para 0000 e o “reset” é ativado.



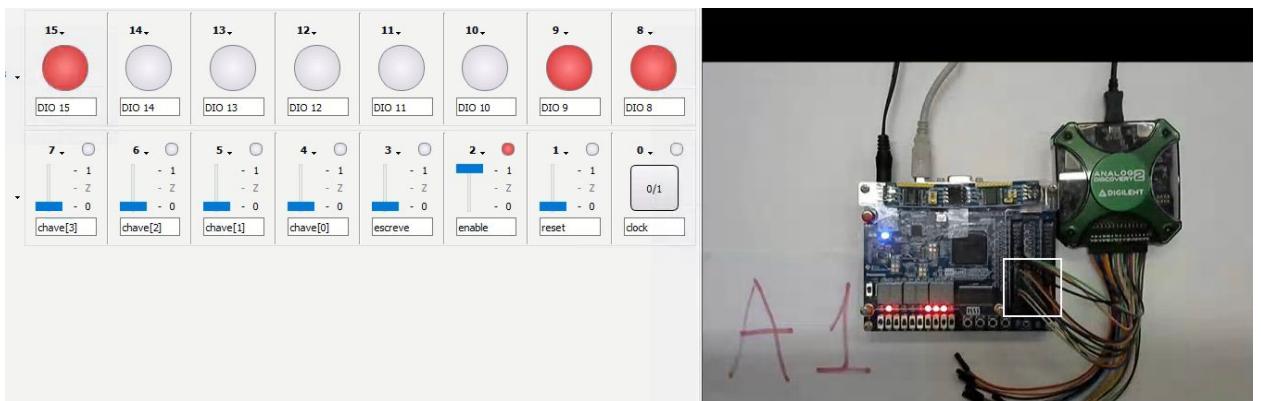
Após 1 pulso de clock, o contador retorna para 0000, e os LEDs que indicam o valor lido na memória nos mostram o valor 0001 novamente.



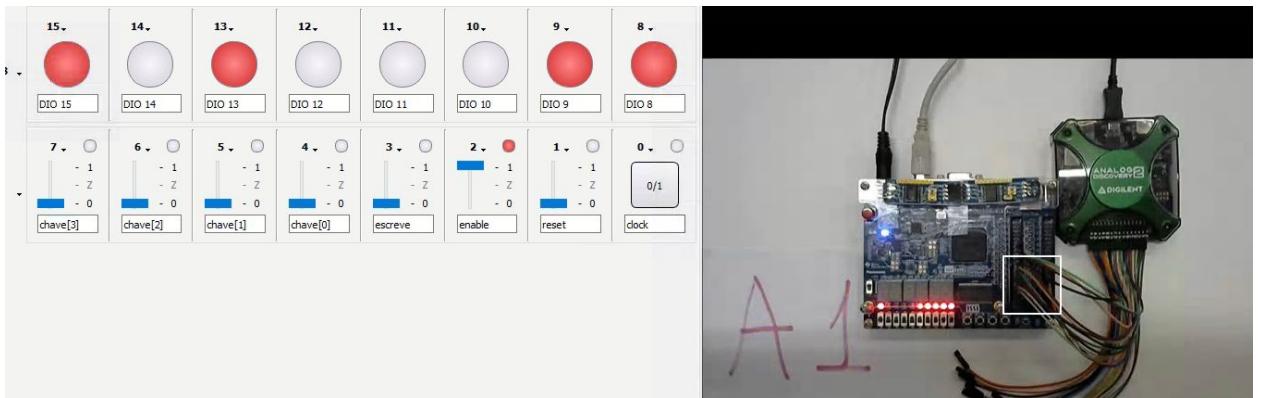
O valor de “chave” foi setado para 1111, o de “escreve” para 1 e o de “reset” para 0.



Após 1 pulso de clock, vemos que o valor 1111 foi escrito na posição 0000 da memória. O LED “igual” acende, pois o valor lido na memória é igual ao de “chave”.

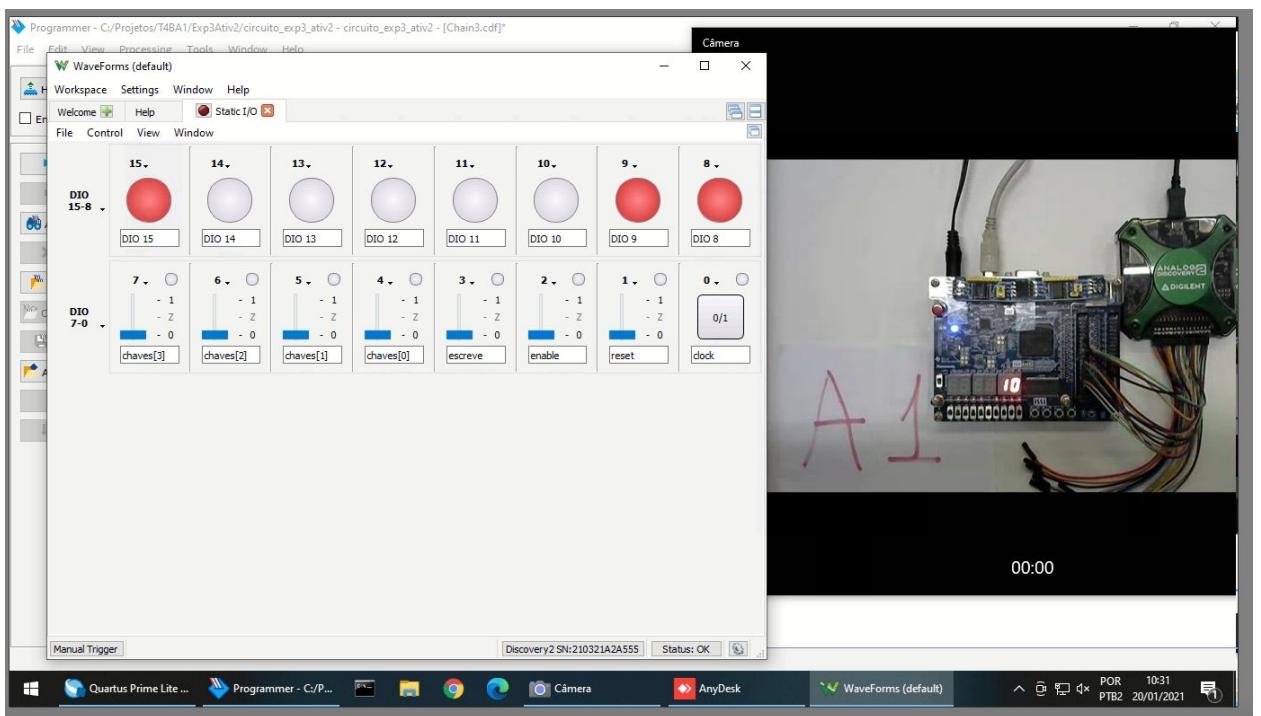


Setamos o valor de “chave” para 0000, o de “escreve” para 0 e o de “enable” para 1 e demos 14 pulsos de clock. Os LEDs do contador indicam o valor 1110 e os da memória indicam o valor 1000.

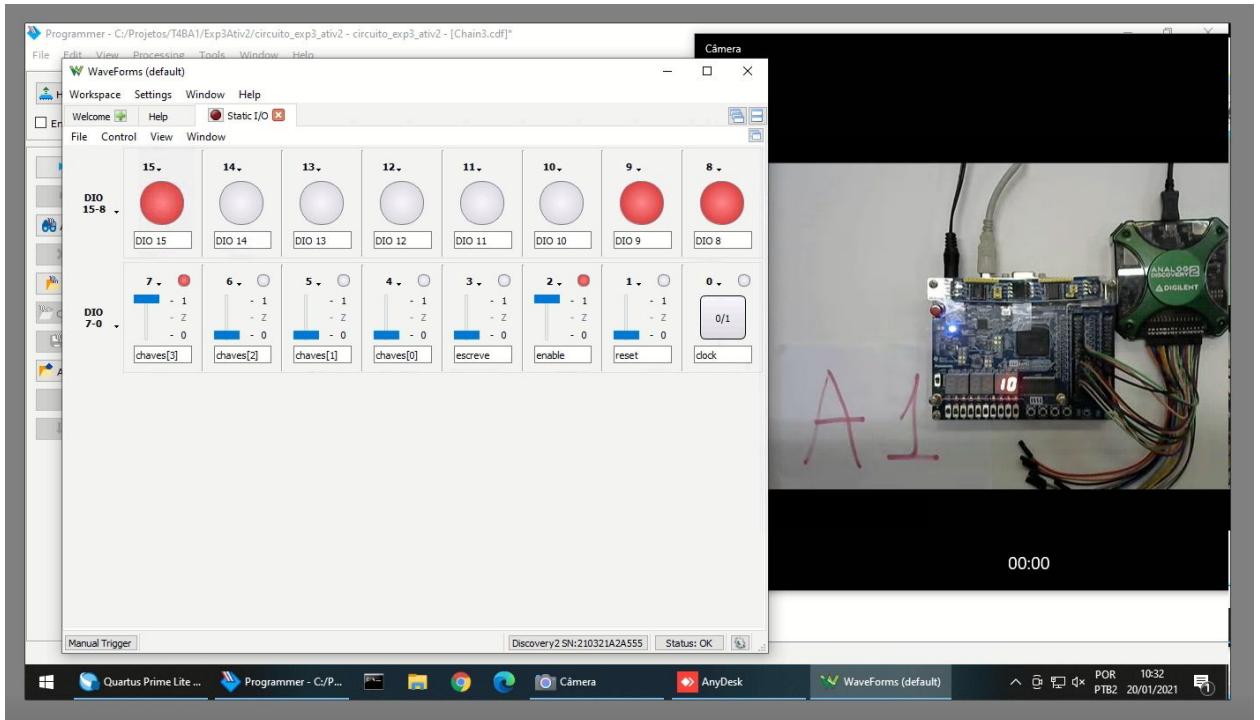


Após mais um pulso de clock, o valor do contador é 1111, o lido na memória é 0000 e o LED que nos mostra o valor de “fim” acende, indicando o fim da contagem.

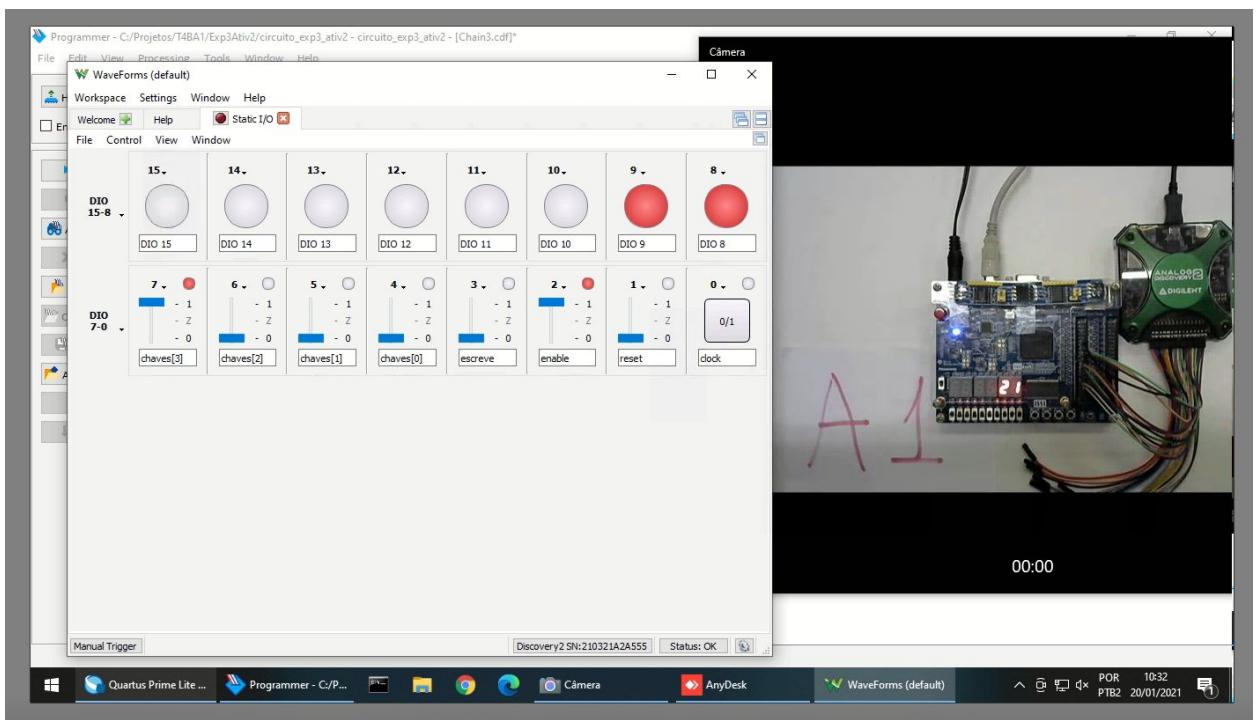
## 6.2) Circuito “circuito\_exp3\_ativ2”



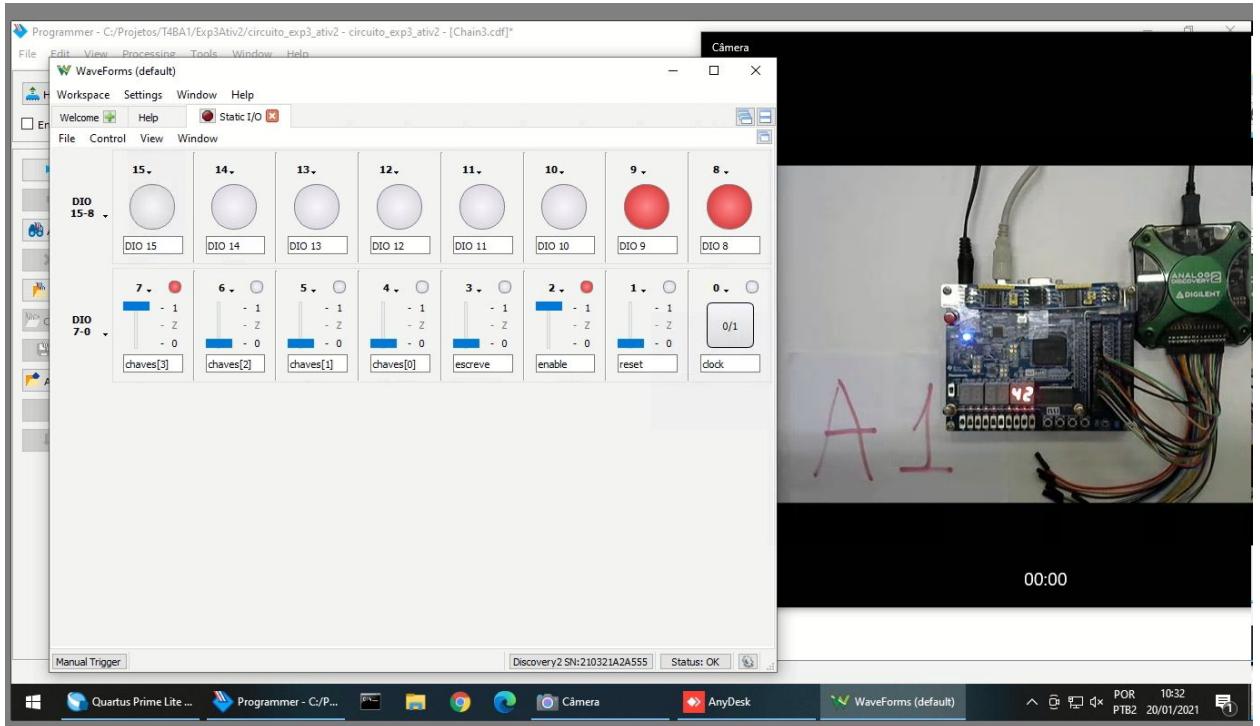
Condições iniciais: contador zerado, memória na posição inicial. O display mais à direita é do contador, e o da esquerda do conteúdo da memória.



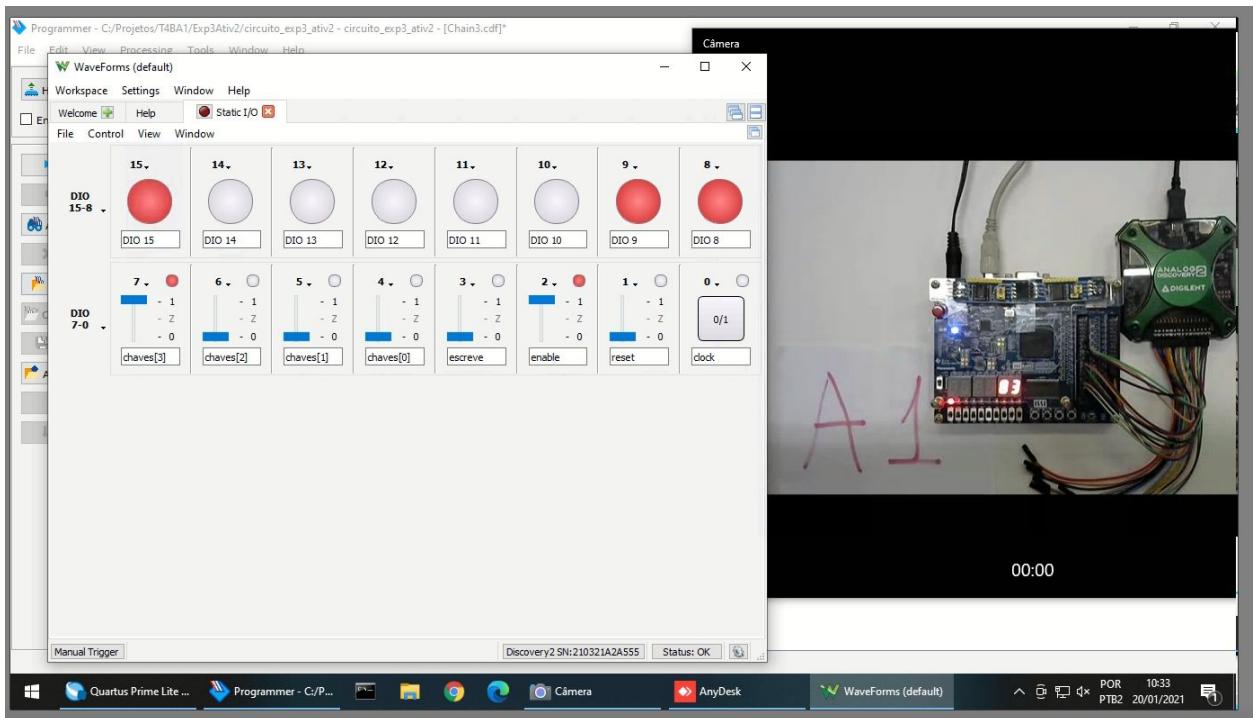
Caso 1: chaves =1000, enable =1



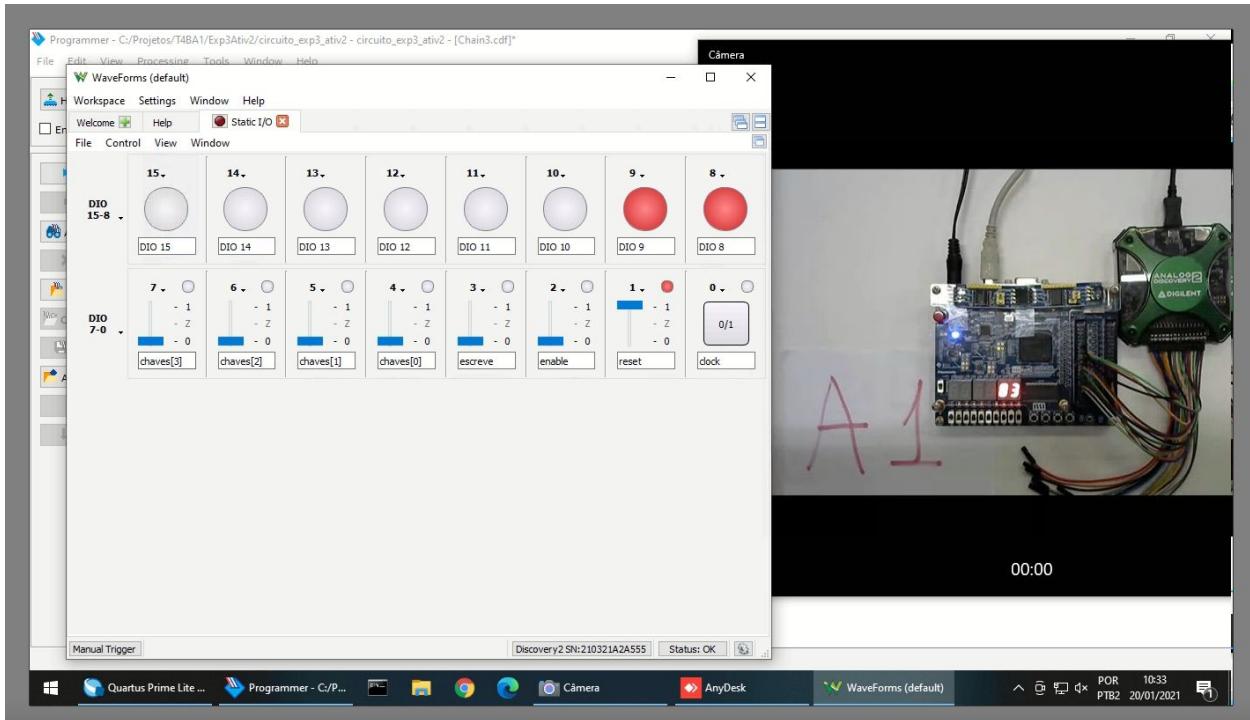
Caso 2: apôs um pulso de clock o contador vai para 1 e a memória para a posição 1, onde está armazenado o valor 2.



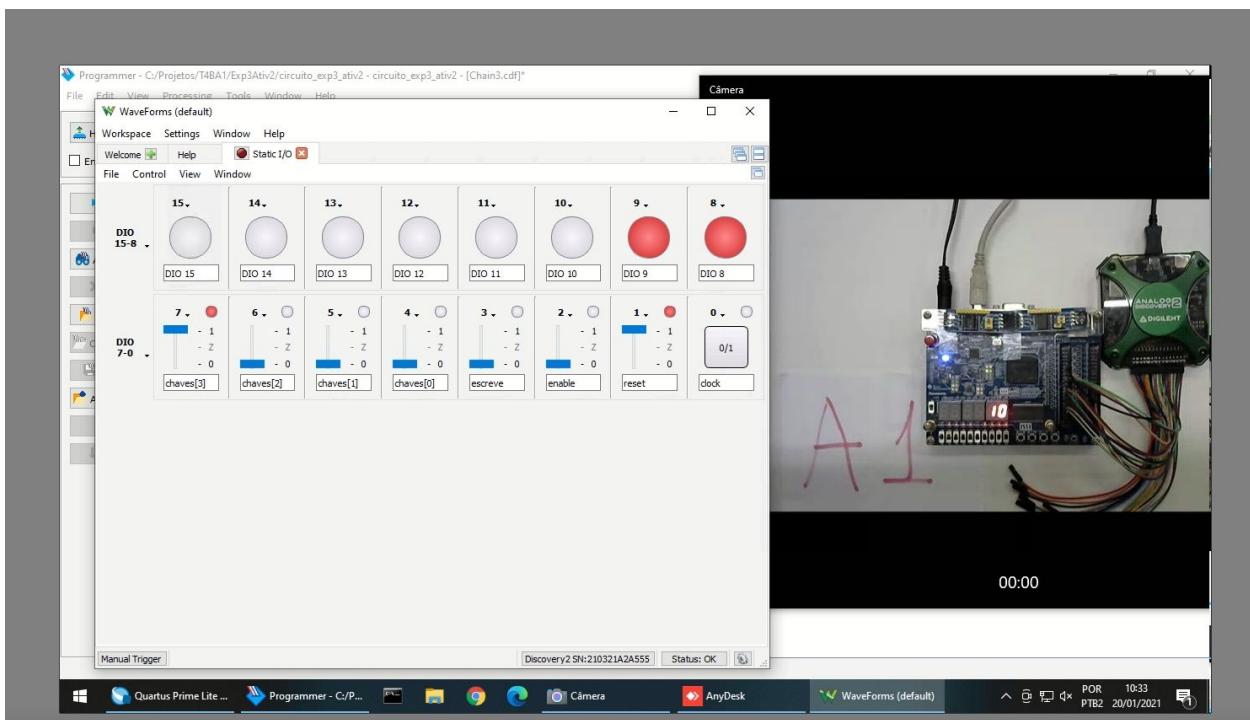
Caso 3: após um pulso de clock é incrementado um endereço de memória e 1 no contador.



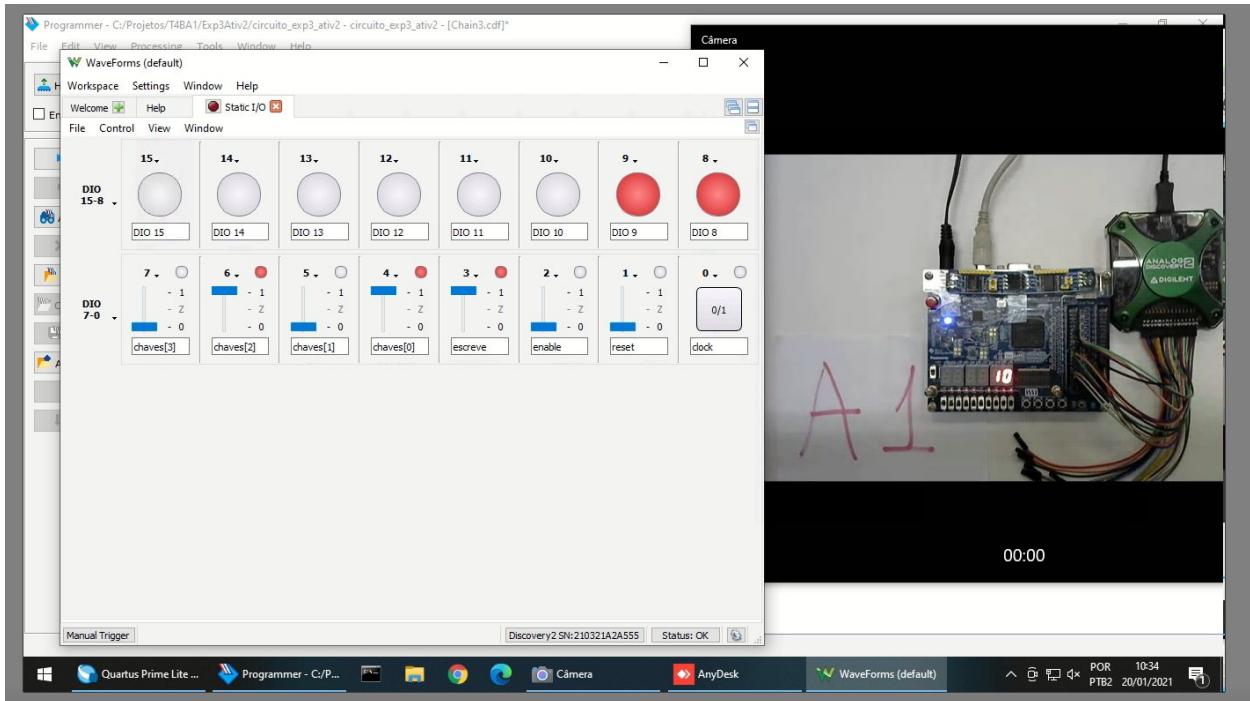
Caso 4: o contador chega a 3, endereço onde está armazenado o valor 8 (1000) que é o mesmo das chaves, acendendo o LED de igual.



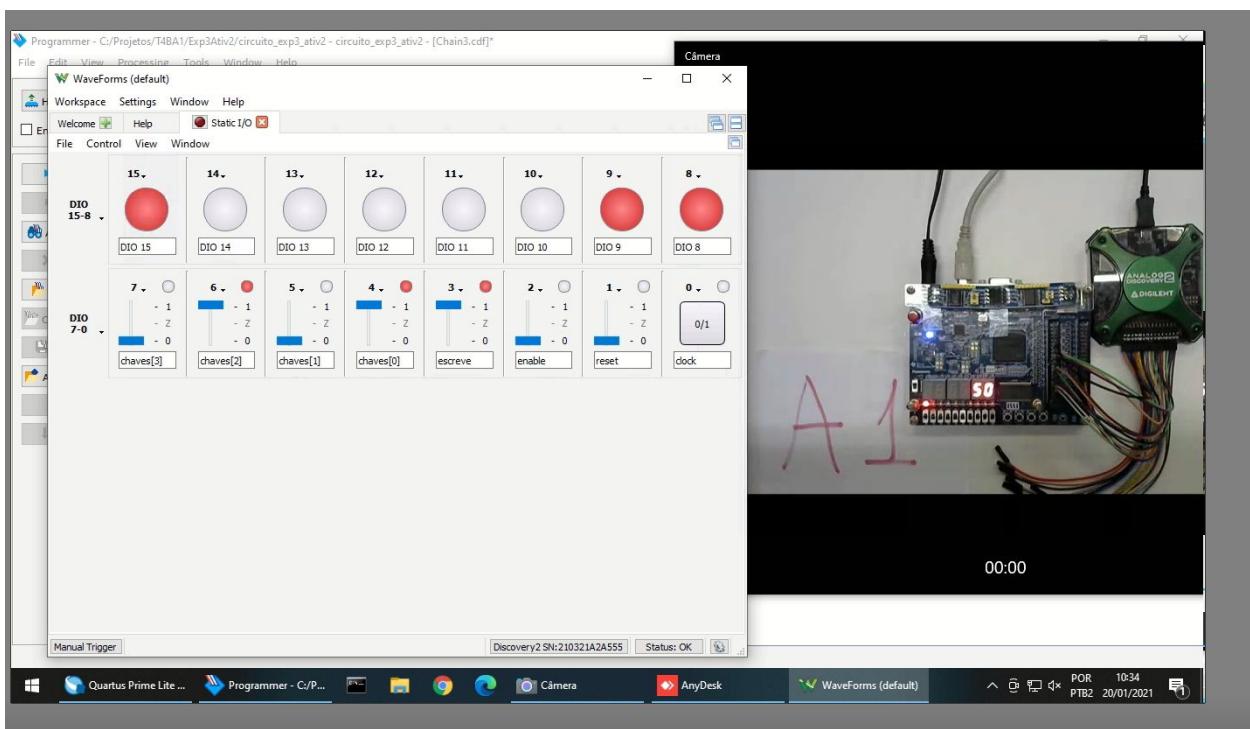
Caso 5: reset habilitado



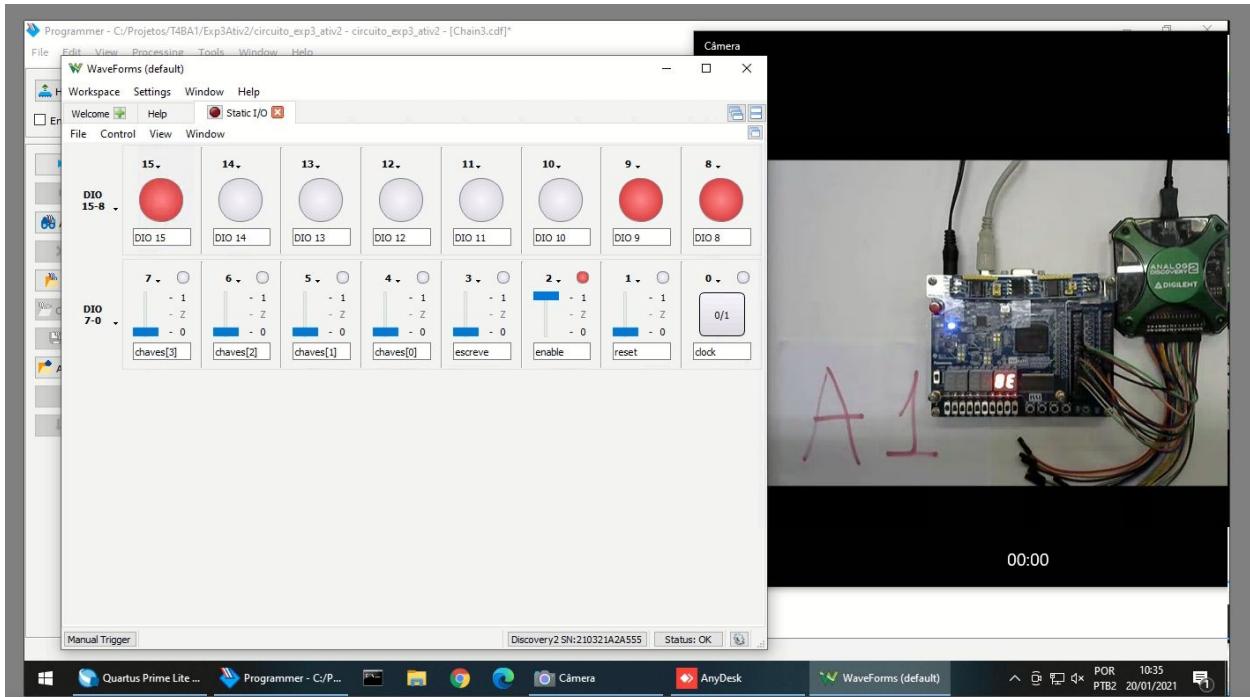
Caso 6: após um clock o contador é zerado



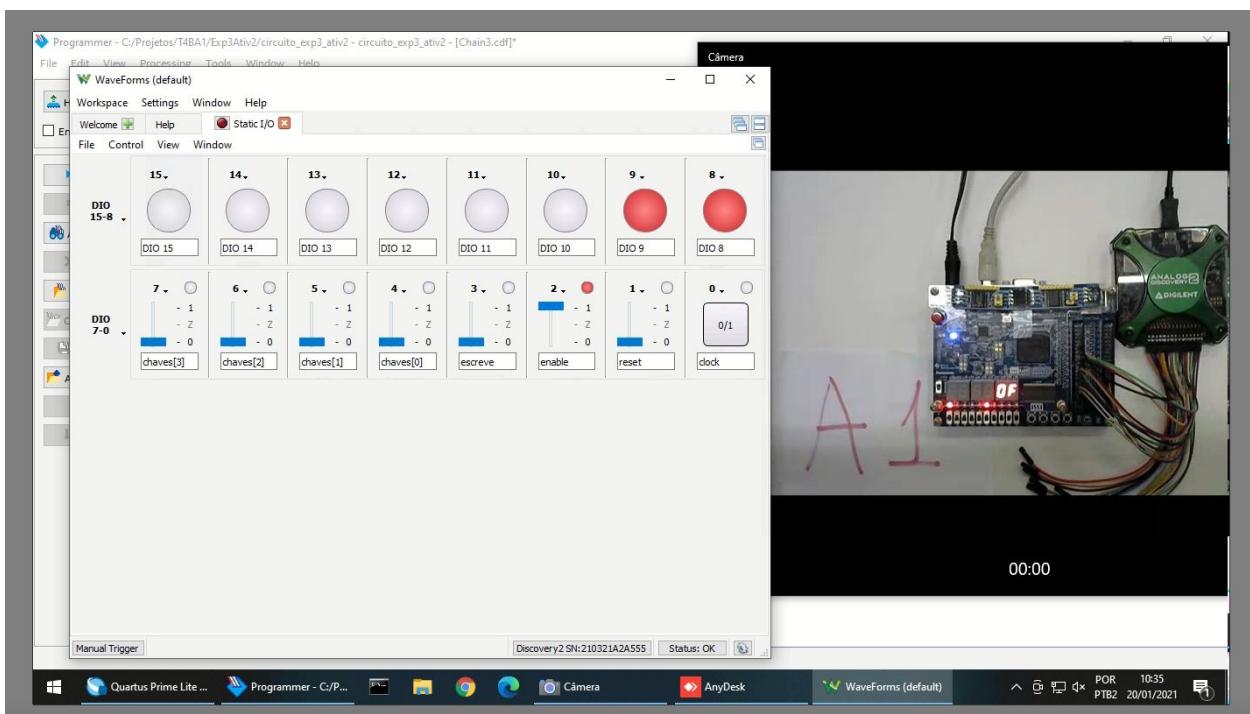
Caso 7: reset desabilitado, chaves = 0101, escreve habilitado



Caso 8: após um período de clock, o número 5 (0101) é escrito na posição 0 da memória

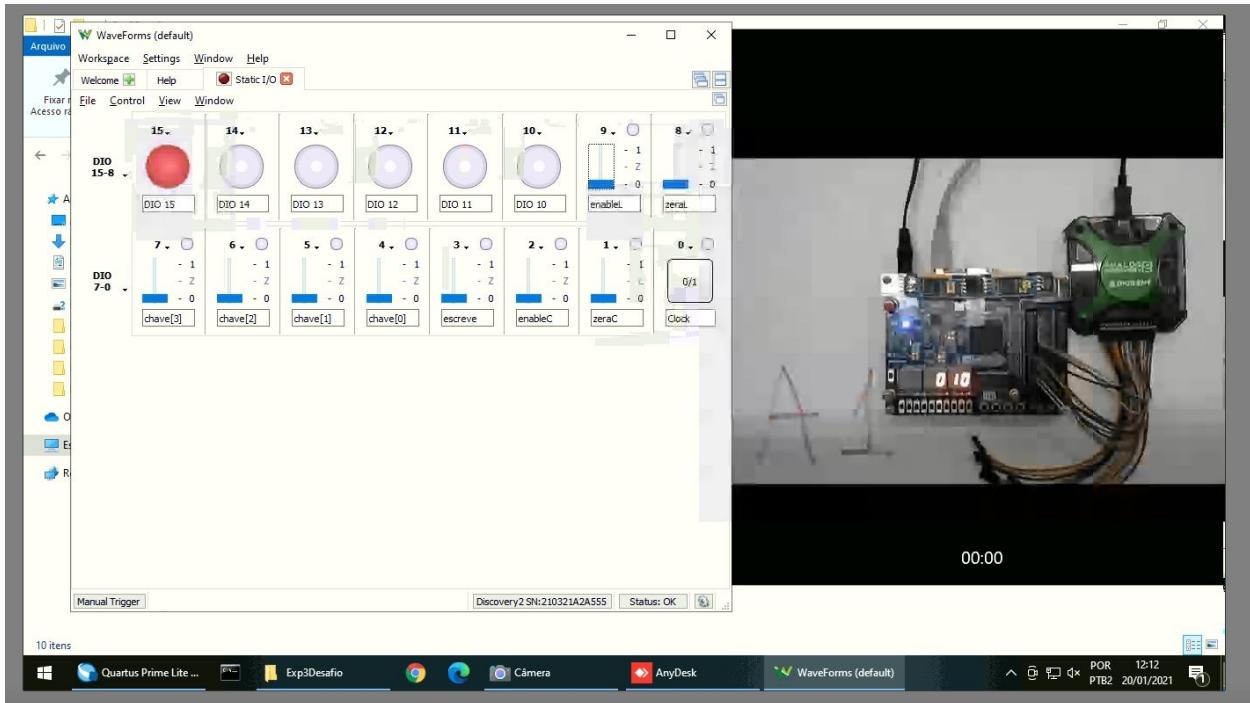


Caso 9: Contador na 14a posição (E) enable habilitado.

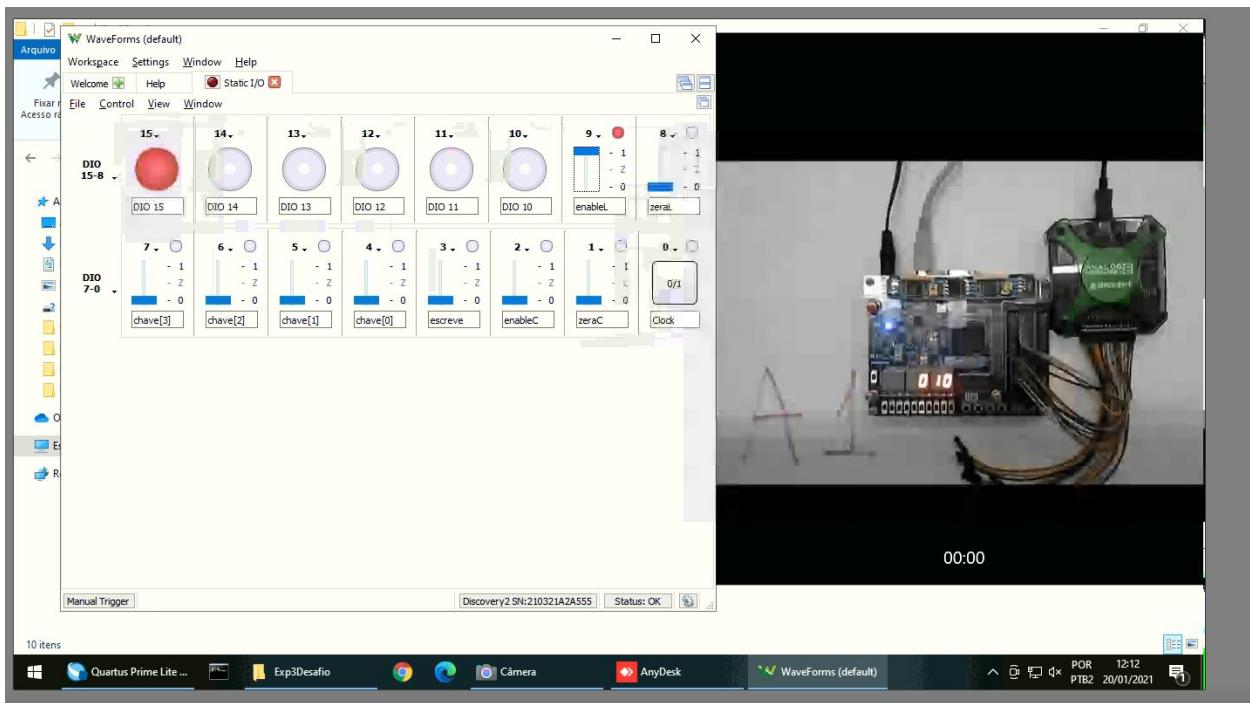


Caso 10: Após um clock o contador chega ao valor 15 (F) e acende o LED de fim. ( o led de igual está aceso pois 0 está escrito na posição 15 da memória o que é igual à chave)

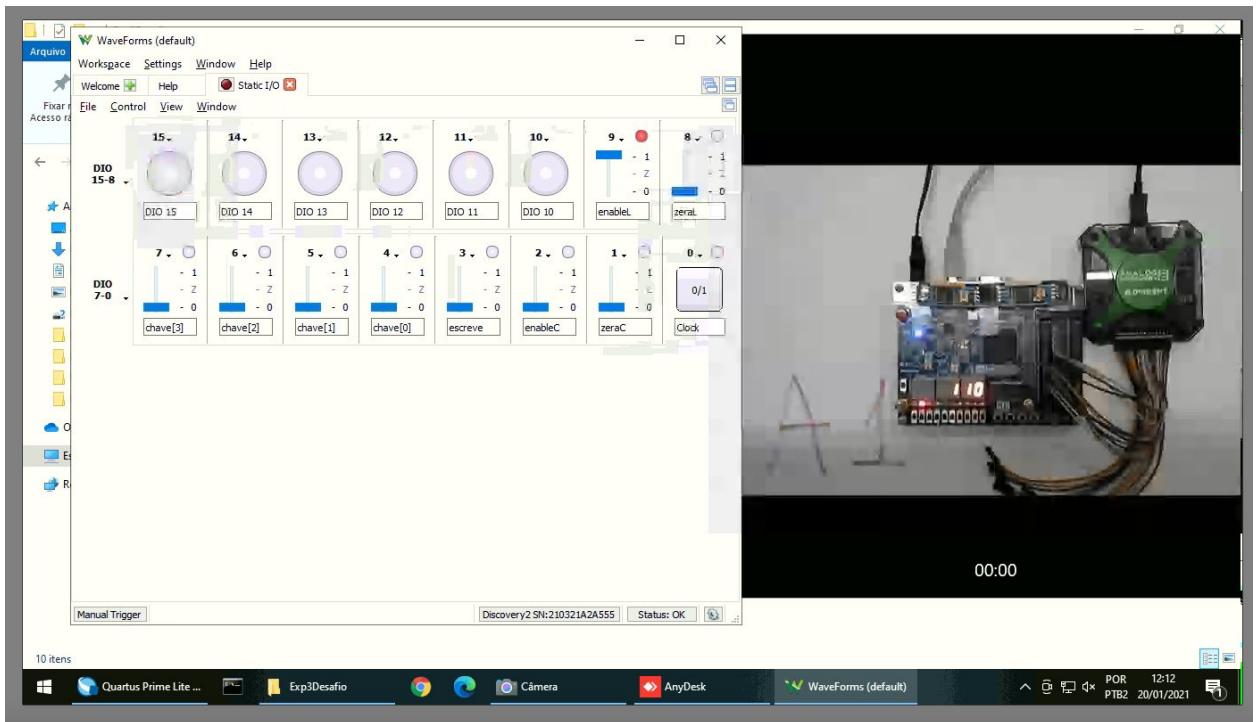
### 6.3) Circuito “circuito\_exp3\_desafio”



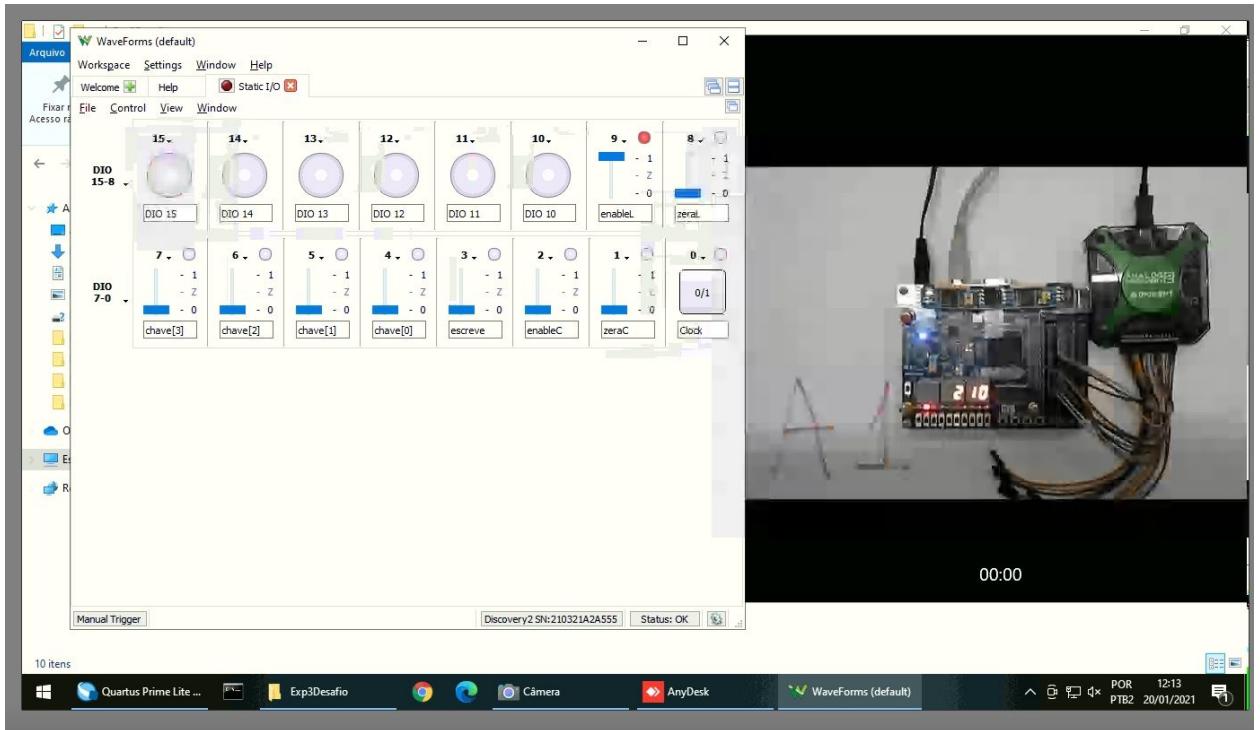
Condições iniciais: contadores zerados. O display mais à esquerda mostra o contador limite, o do meio o conteúdo da memória e o mais à direita é o contador de endereços



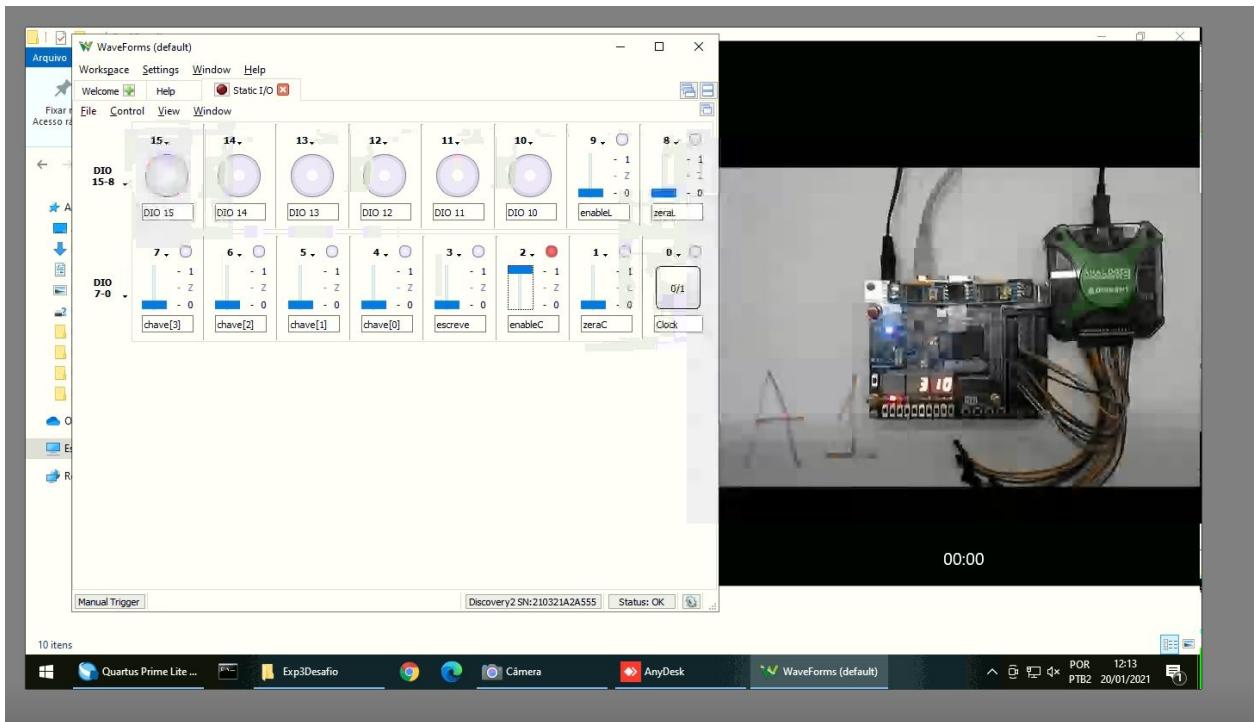
Caso 1: enable do contador limite habilitado



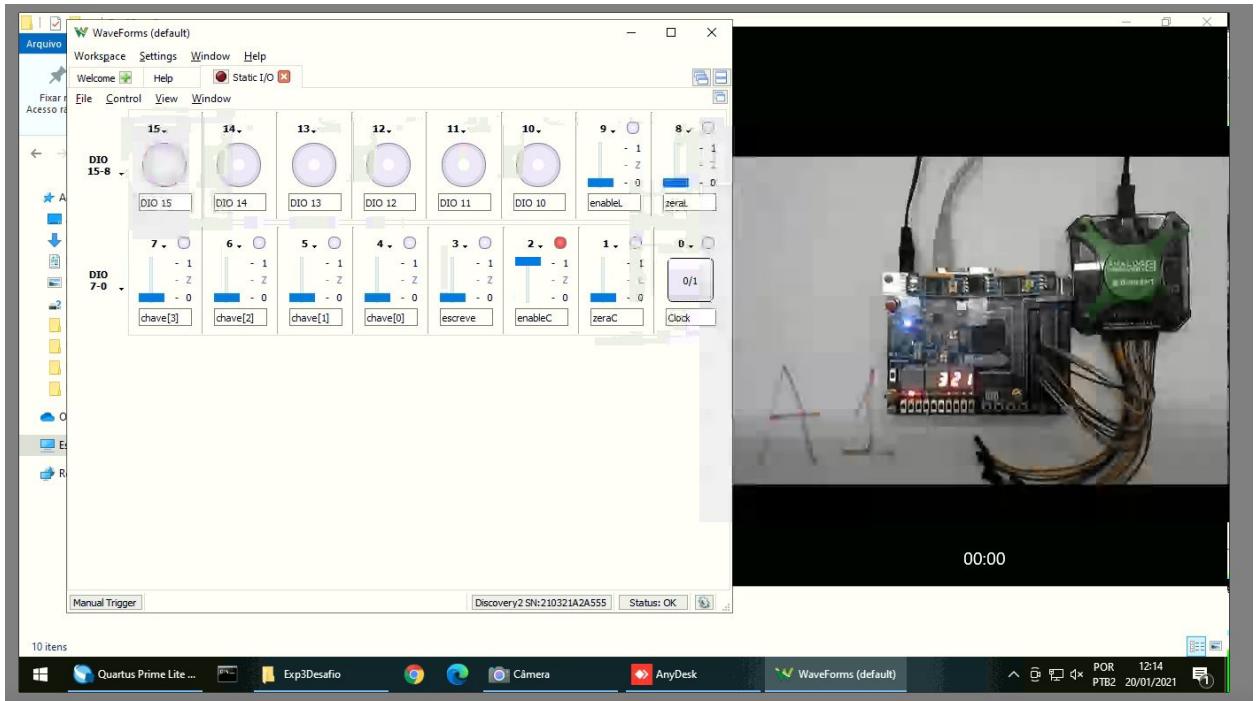
Caso 2: Após um clock o contador limite é acrescentado 1. O LED de endereço menor que limite acende pois o contador de endereços está numa posição menor (0)



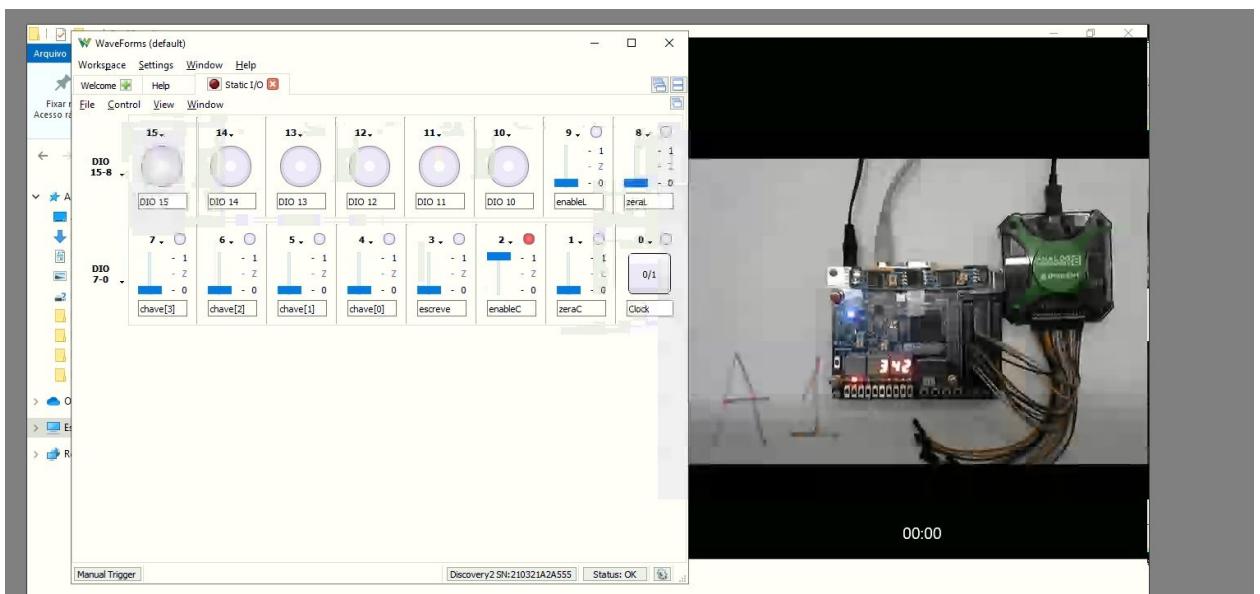
Caso 3: Após um clock o contador limite é acrescentado 1



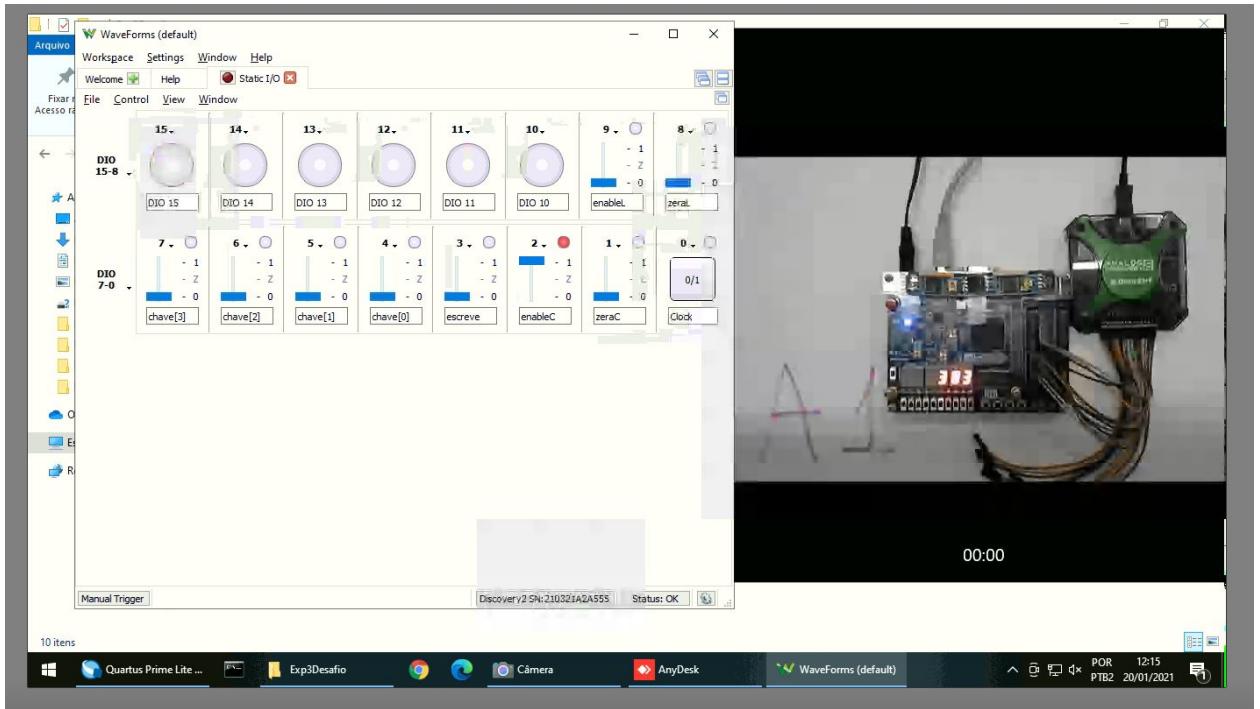
Caso 4: Após um clock o contador limite é igual a 3. O enable do contador limite é desligado e o do contador de endereços é ligado.



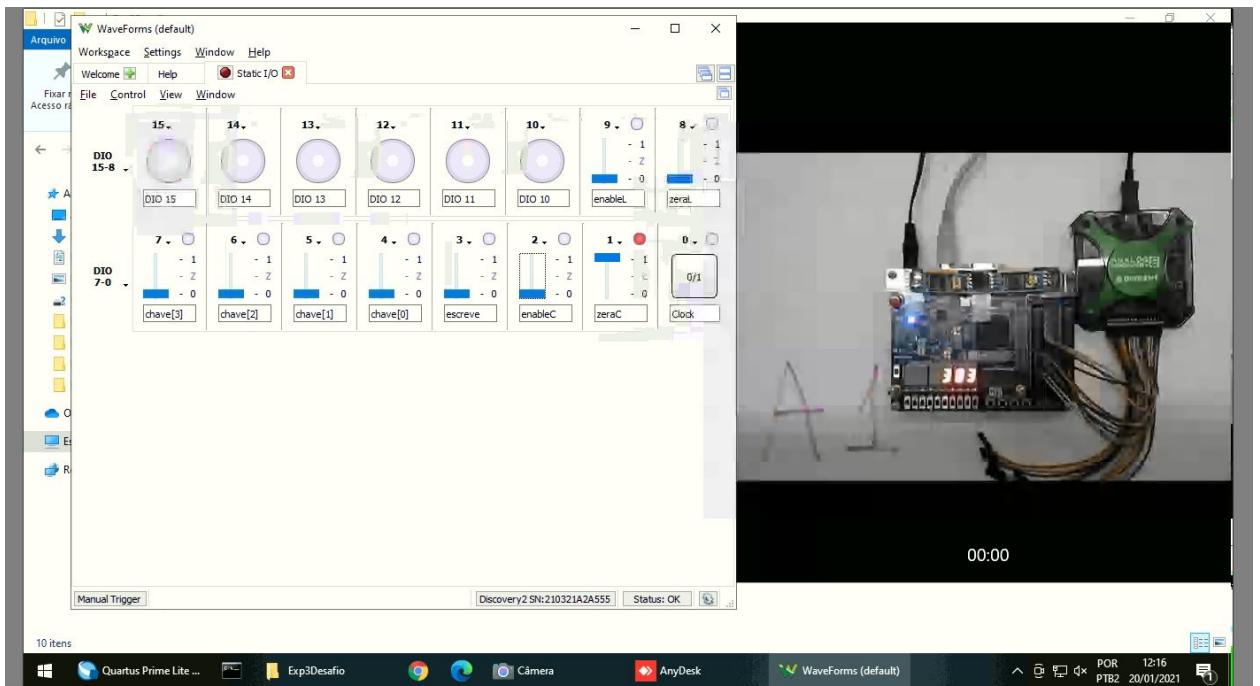
Caso 5: após um clock, o contador de endereços é incrementado em um



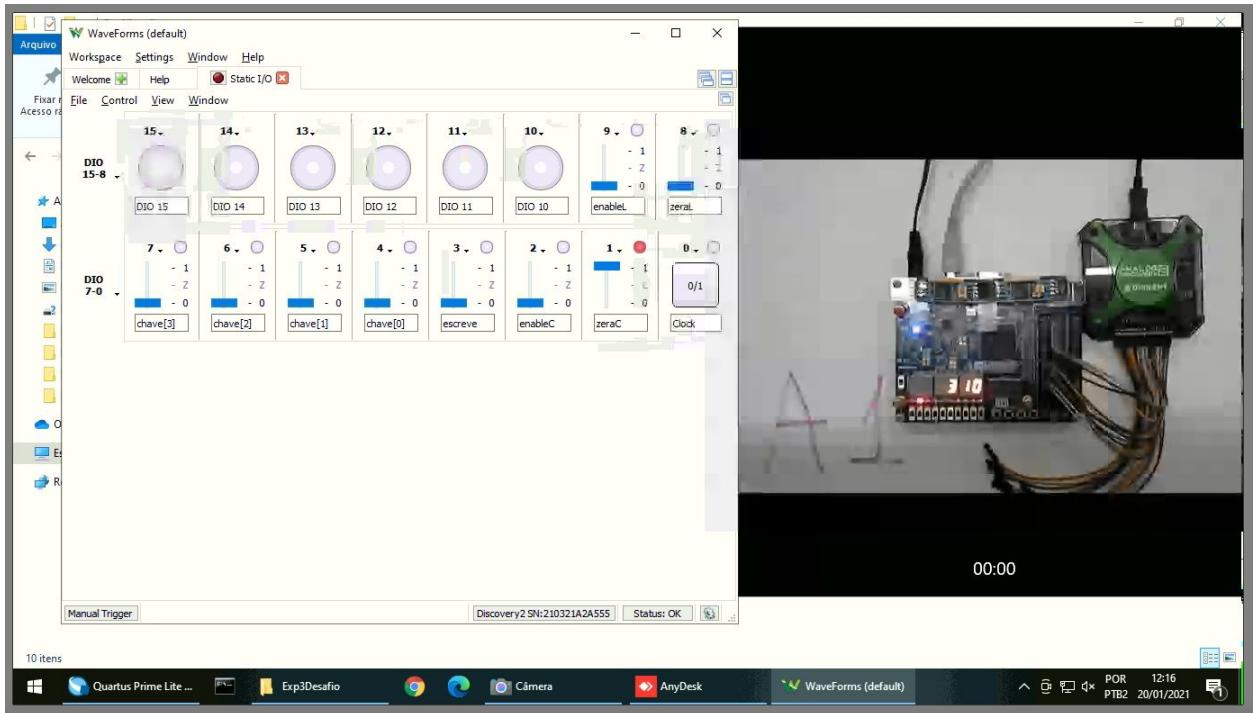
Caso 6: após um clock, o contador de endereços é incrementado em um



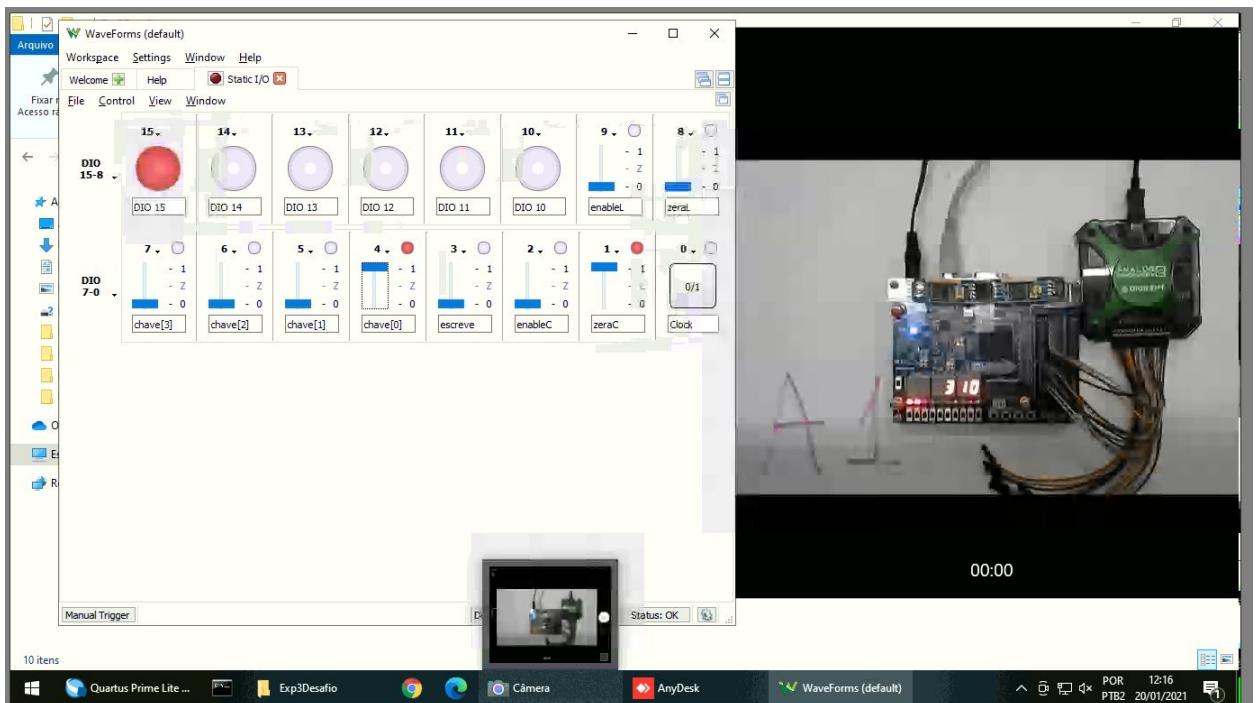
Caso 7: após um clock, o contador de endereços é incrementado em um. O led de endereço menor que limite é apagado pois os dois se encontram na mesma posição.



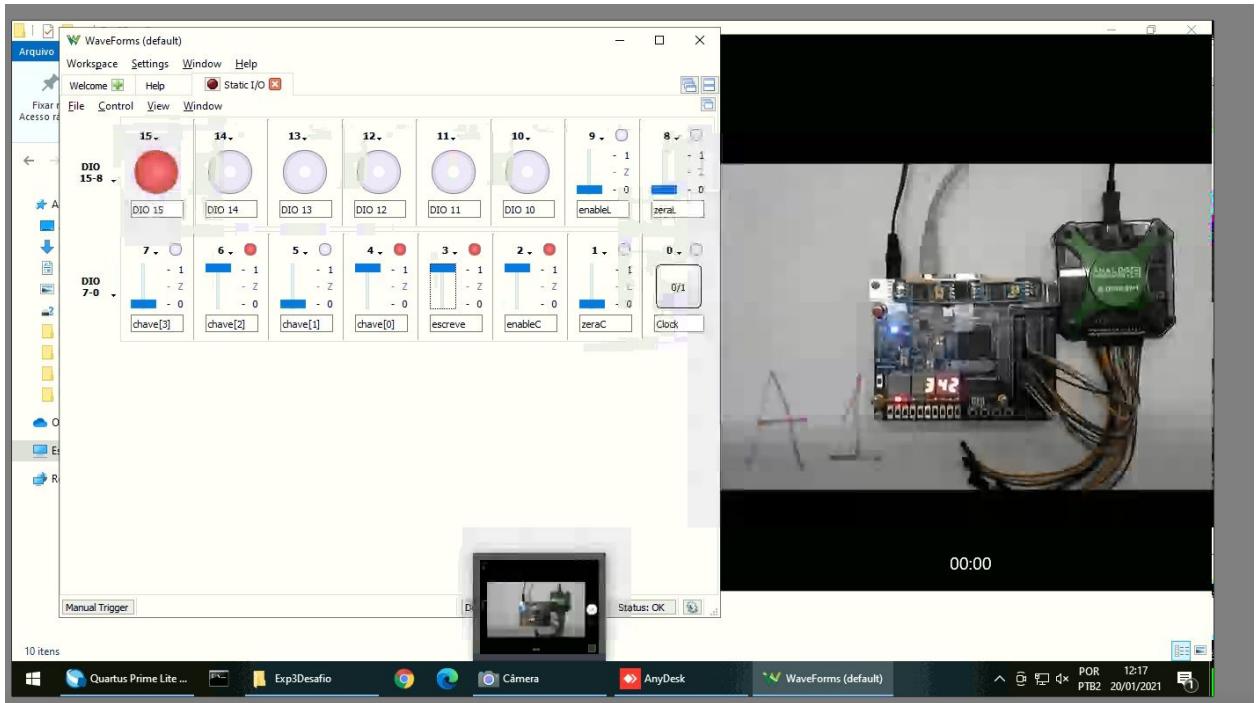
Caso 8: o enable do contador C é desligado, e o zeraC habilitado



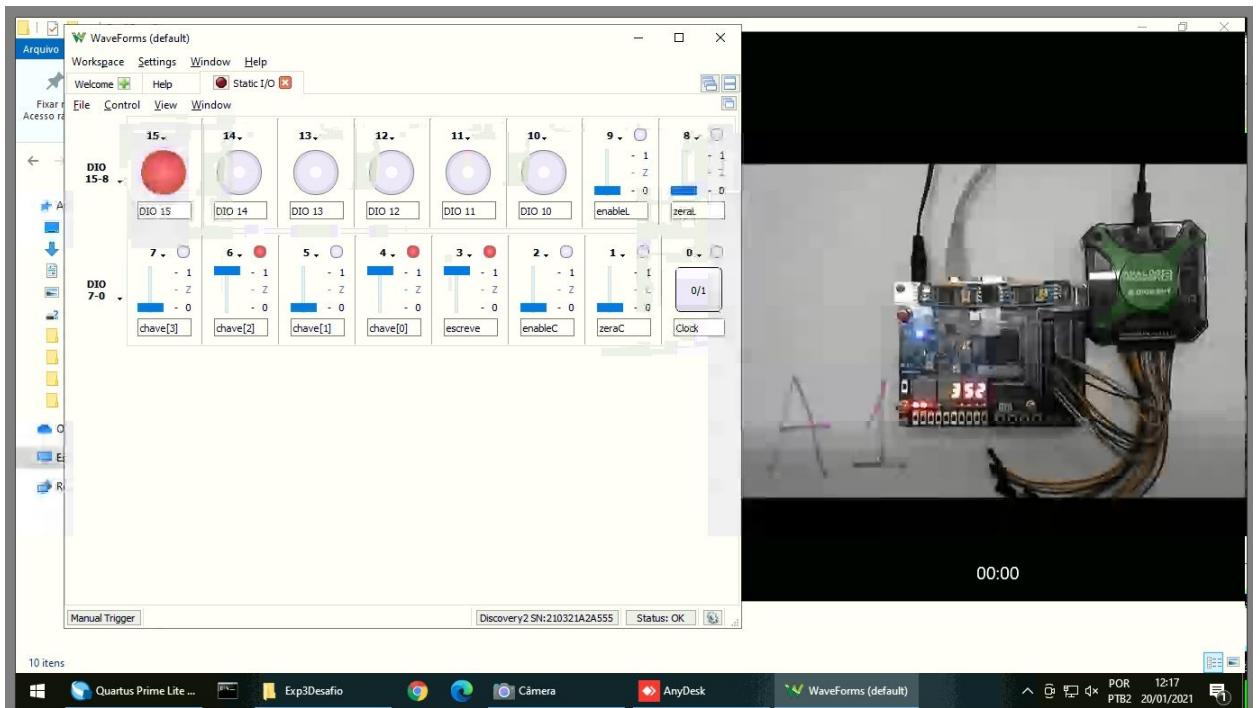
Caso 9: após um clock, o contador de endereços é zerado. O LED de endereço menor que limite é aceso novamente.



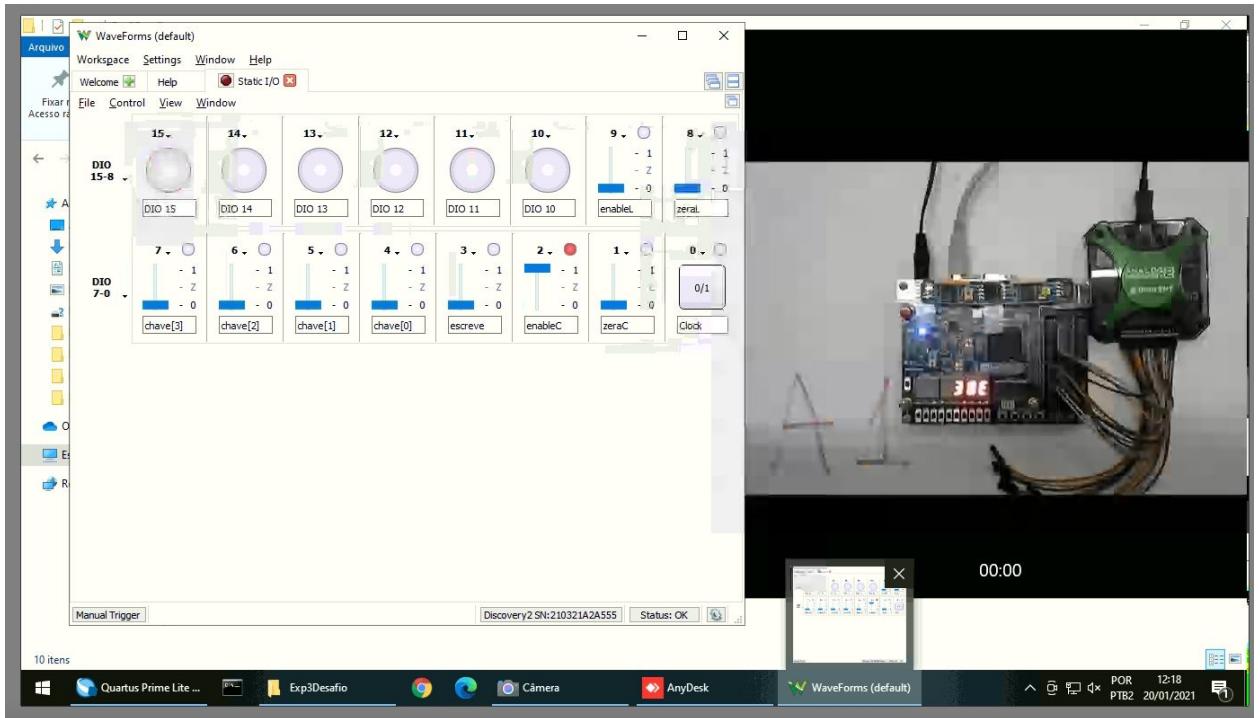
Caso 10: A chave está na posição 0001, que é igual ao conteúdo da memória nessa posição (0), logo, o LED de chaves igual a memória é aceso.



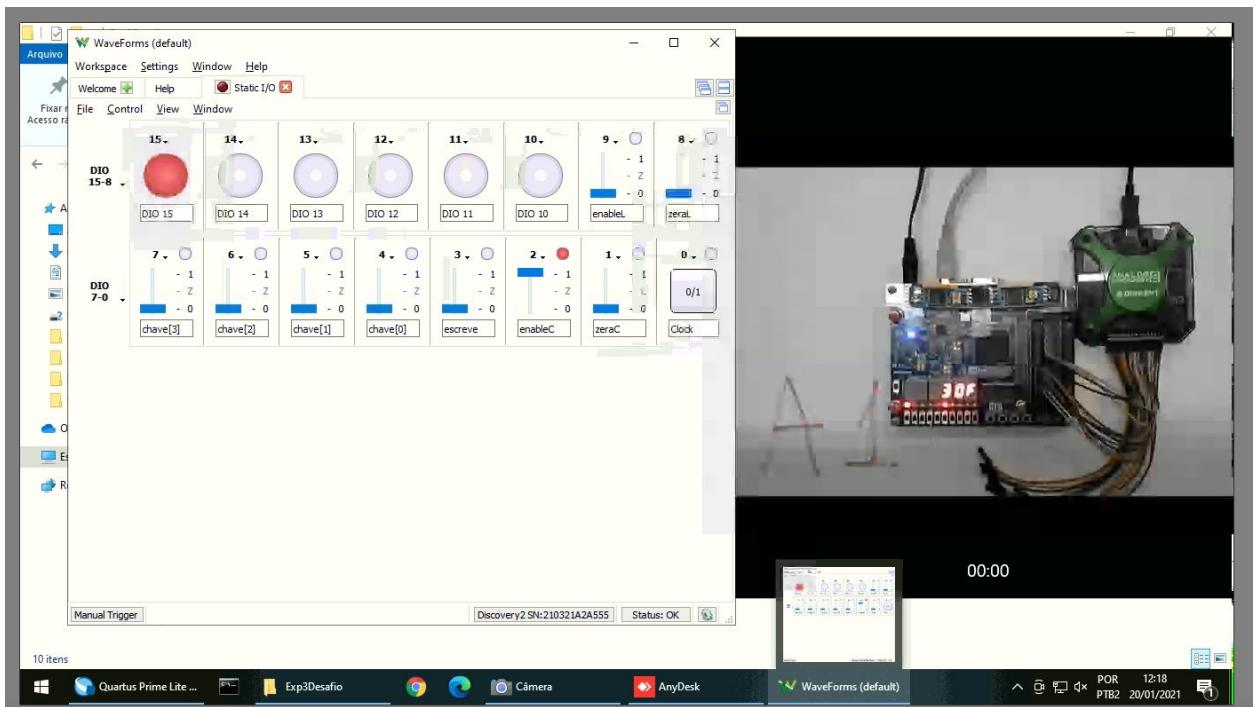
Caso 11: O contador de endereços está na posição 2, as chaves em 0101 e o escreve é habilitado.



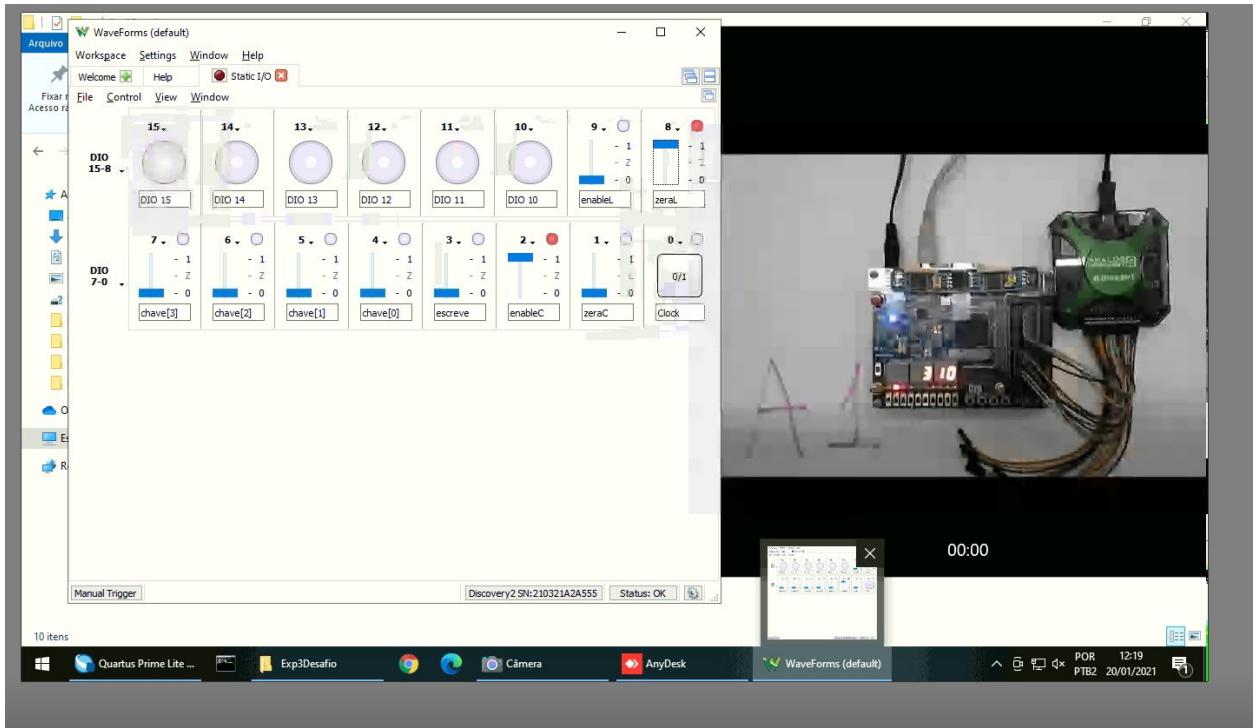
Caso 12: após um clock, o número 5 (0101) é escrito na posição 2 da memória.



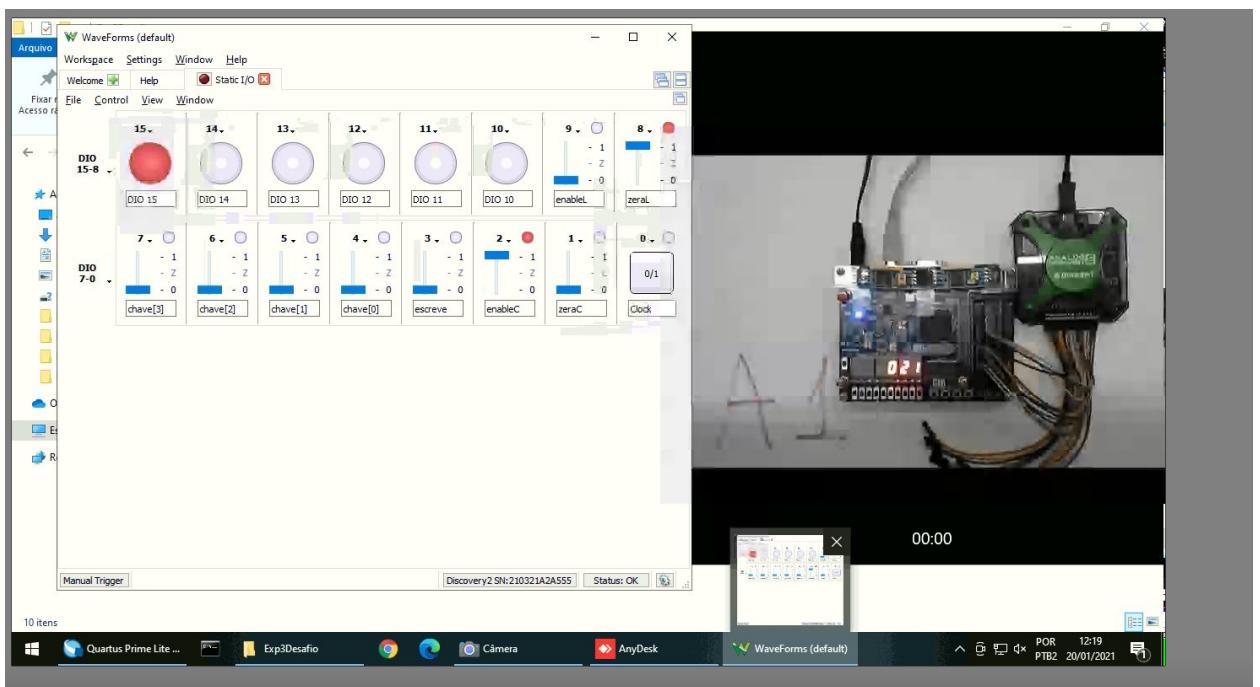
Caso 13: o contador de endereços está na posição 14 (E)



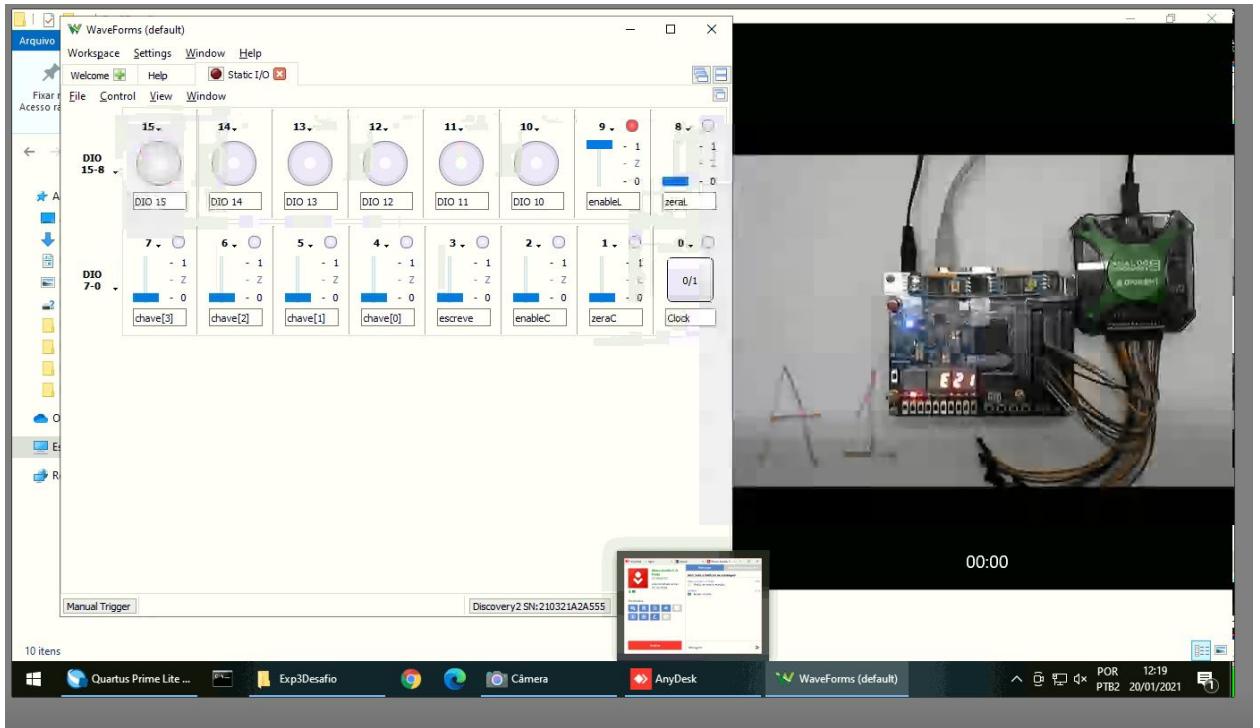
Caso 14: após um clock, o contador de endereços é incrementado em um e vai para 15 (F) o que faz o LED de FimC acender.



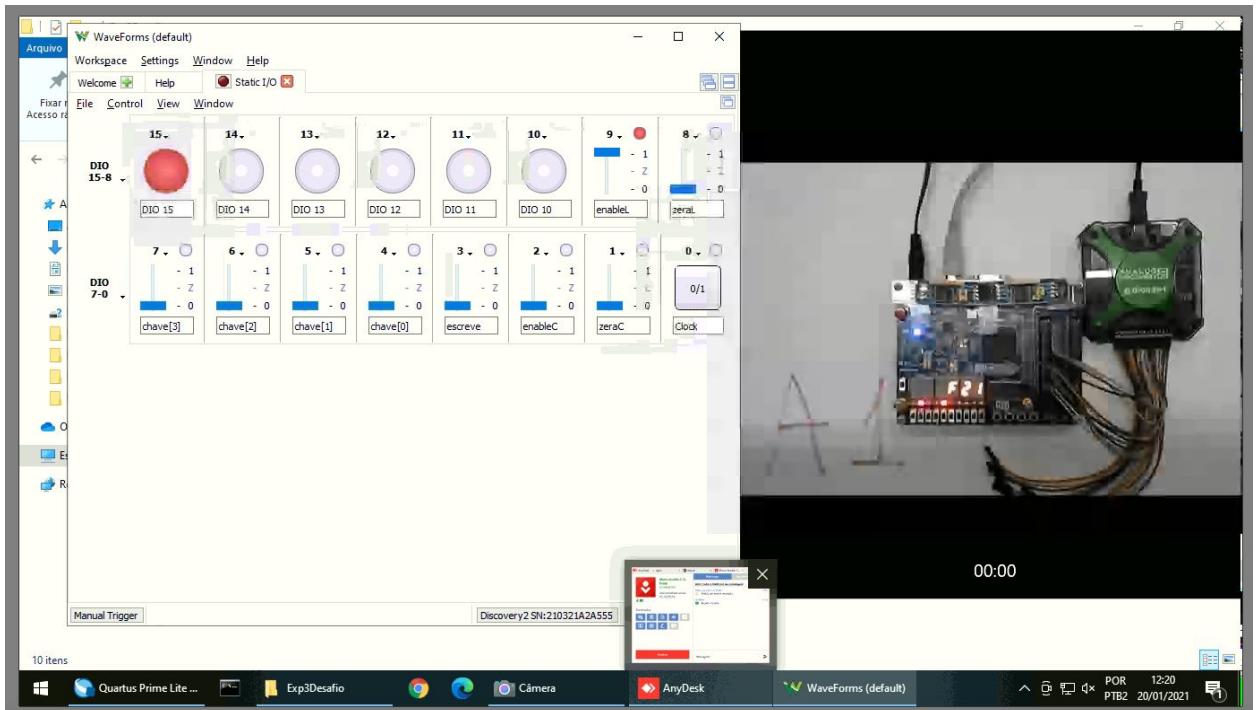
Caso 15: a chave zeraL e o enableC estão habilitados



Caso 16: após um clock, o contador de endereços é incrementado em um e o contador limite é zerado.



Caso 17: o contador limite está na posição 14 (E)



Caso 18: após um clock, o contador limite é incrementado para a posição 15 (F) o que faz o LED FimL acender.

## 7) Resultados Alcançados

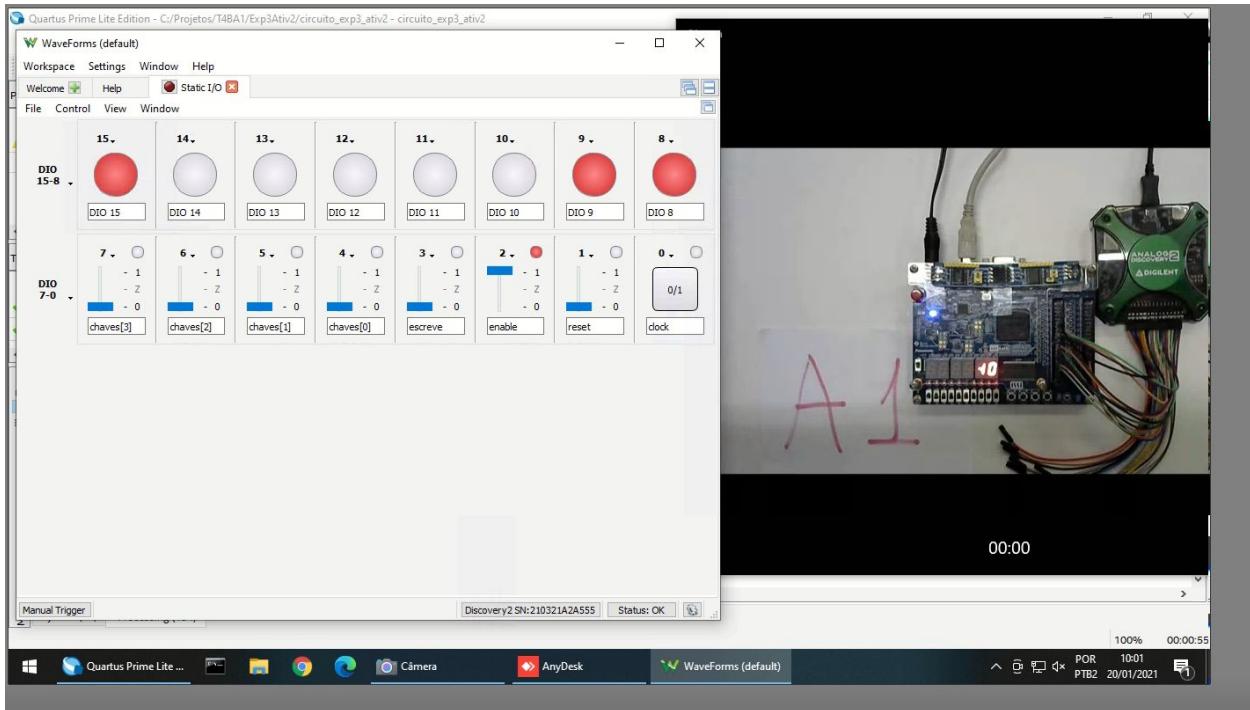
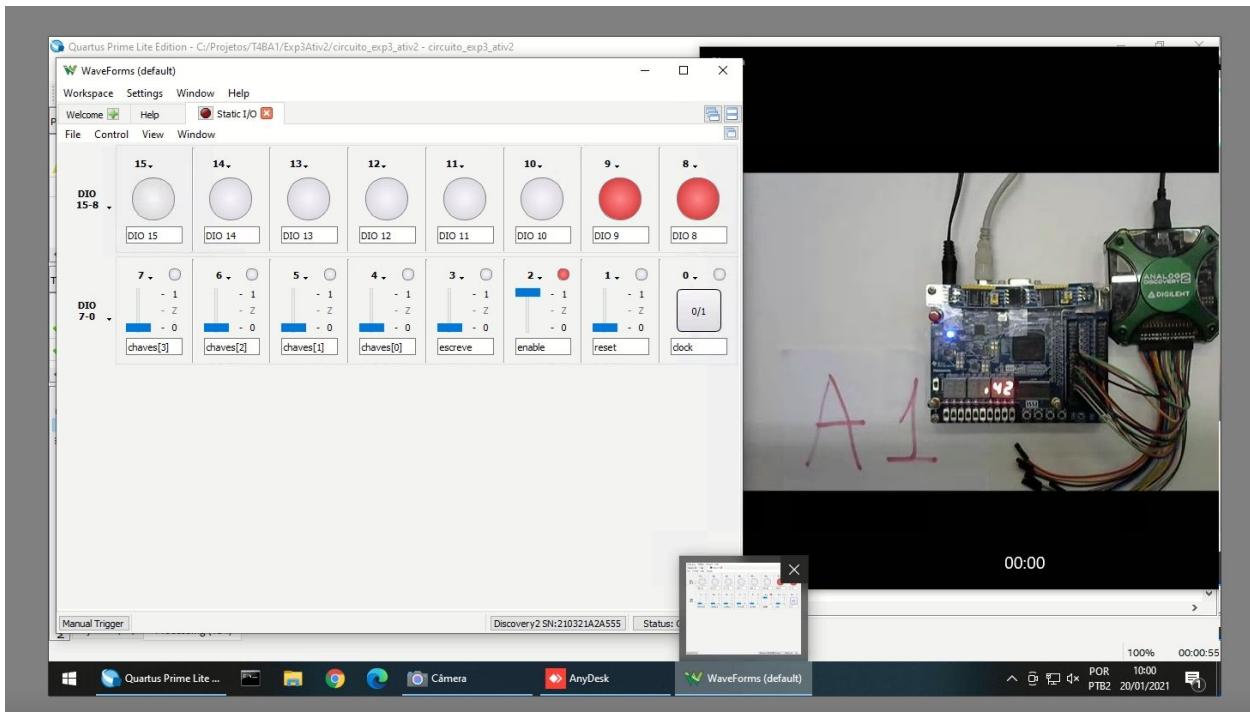
## **7.1) Pontos Positivos**

Em geral, a experiência correu bem e as dificuldades que surgiram foram rapidamente contornadas. Nosso planejamento se mostrou efetivo e conseguimos realizar todas as atividades propostas.

## **7.2) Pontos Negativos**

Por conta da versão do Quartus instalada nos computadores do Laboratório Digital, foi notado um comportamento síncrono nas memórias utilizadas nos circuitos desenvolvidos, que originalmente deveriam ser assíncronas. Fomos alertados sobre esse problema pelo professor Edson Midorikawa antes da realização do experimento e, portanto, ele foi facilmente contornado através do uso do clock nas operações que envolviam a memória.

Tivemos também um problema relacionado à pinagem do circuito “circuito\_exp3\_ativ2”, que foi planejado com um leve erro. Este erro fez com que a exibição dos valores da memória não ocorresse de maneira adequada. O erro pode ser visualizado nas imagens abaixo:



### 7.3) Lições Aprendidas

Através deste experimento, pudemos ver na prática o uso e o funcionamento das memórias e dos displays de 7 segmentos em circuitos digitais, consolidando assim, conhecimentos adquiridos nas disciplinas Sistemas Digitais I e Sistemas Digitais II. Além disso, reforçamos a boa prática de testes e

planejamento, pois como havíamos simulado o circuito 2 com antecedência, pudemos facilmente localizar o erro como sendo a pinagem, pois a simulação havia dado certo.

## **8) Referências Técnicas**

1. Apostila e apresentação da Experiência 3, disponibilizados no ambiente da disciplina PCS3635 no site E-disciplinas.
2. Tabela de Pinos da placa DE0-CV
3. Vídeo de apresentação da Experiência 3 gravado pelo professor Edson Midorikawa