



**Laboratório Digital I - PCS3635**

**Planejamento da Experiência 2:**

**Um Fluxo de Dados Simples**

Marco Aurélio C. O. Prado - NUSP 11257605

Victor Hoefling Padula - NUSP 10770051

Turma 04 - Bancada A1

São Paulo - SP

13/01/2021

## **1) Objetivo**

Nesta experiência, visa-se estudar, simular e testar em uma FPGA um circuito comparador de magnitude.

Após isso, será projetado um novo circuito utilizando o comparador fornecido e o contador que utilizamos na Experiência 1. Este novo circuito passará pelo mesmo processo de estudo, simulação e testes em FPGA.

## **2) Estudo da descrição VHDL fornecida**

### **2.1) Sinais dos pinos do circuito integrado e suas respectivas funções**

### **2.2) Perguntas a respeito do circuito fornecido**

#### **2.2.1) Qual é o intervalo de valores possíveis das entradas A e B?**

De 0 a 15, visto que A e B possuem 4 bits. É possível aumentar o intervalo de valores possíveis cascadeando dois ou mais comparadores.

#### **2.2.2) Este componente é sensível a qual borda do sinal de clock?**

Não, visto que não há entrada de clock no componente.

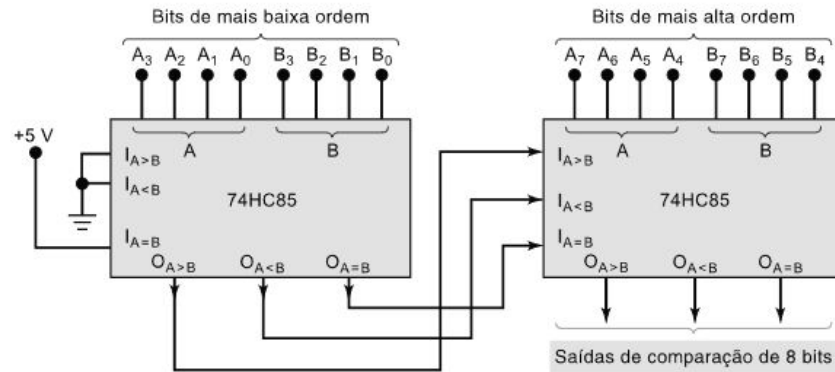
#### **2.2.3) Qual é a função das entradas $A > B$ , $A = B$ e $A < B$ ? Que valor essas entradas devem possuir para que o 7485 seja usado para comparar dois valores binários de 4 bits?**

Essas entradas servem para cascadear vários comparadores, expandindo assim o intervalo de valores possíveis de A e B. Para comparar dois valores binários de 4 bits,  $A > B = 0$ ,  $A = B = 1$  e  $A < B = 0$ .

#### **2.2.4) Explique as condições para que a saída $A = B$ out tenha nível lógico alto.**

As condições para que a saída  $a=b$  tenha nível lógico alto são que bit a bit A e B devem seguir a lógica  $\text{not}(a \text{ xor } B)$ , indicando que os bits são iguais, e a entrada  $I\_AEQB$  deve estar em nível lógico alto.

**2.2.5) Mostre em um diagrama esquemático como dois comparadores de 4 bits podem ser cascadeados para formar um comparador de 8 bits.**



Fonte: Material sobre 7485 do livro Sistemas Digitais (Tocci) fornecido no E-disciplinas

### 2.3) Descrição VHDL comentada

```
-----
-- 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
    cascadeamento
    signal agtb : std_logic;
    signal aeqb : std_logic;
    signal altb : std_logic;
begin
    -- equacoes dos sinais: pagina 462, capitulo 6 do livro-texto
    -- Wakerly, J.F. Digital Design - Principles and Practice, 4th Edition
    -- veja tambem datasheet do CI SN7485 (Function Table)
    agtb <= (i_A3 and not(i_B3)) or
        (not(i_A3 xor i_B3) and i_A2 and not(i_B2)) or
        (not(i_A3 xor i_B3) and not(i_A2 xor i_B2) and i_A1 and not(i_B1)) or
        (not(i_A3 xor i_B3) and not(i_A2 xor i_B2) and not(i_A1 xor i_B1) and i_A0 and
not(i_B0));
    -- checa se a > b
    aeqb <= not(((i_A3 xor i_B3) or (i_A2 xor i_B2) or (i_A1 xor i_B1) or (i_A0 xor i_B0)));
    -- checa se a = b
    altb <= not(agtb or aeqb);
    -- checa se a < b
    o_AGTB <= agtb or (aeqb and (not(i_AEQB) and not(i_ALTB)));
    o_ALTB <= altb or (aeqb and (not(i_AEQB) and not(i_AGTB)));
    o_AEQB <= aeqb and i_AEQB;
    -- nas saidas, são levadas em consideracao as entradas de cascadeamento para obter um
    resultado final

end architecture dataflow; -- fim da arquitetura

```

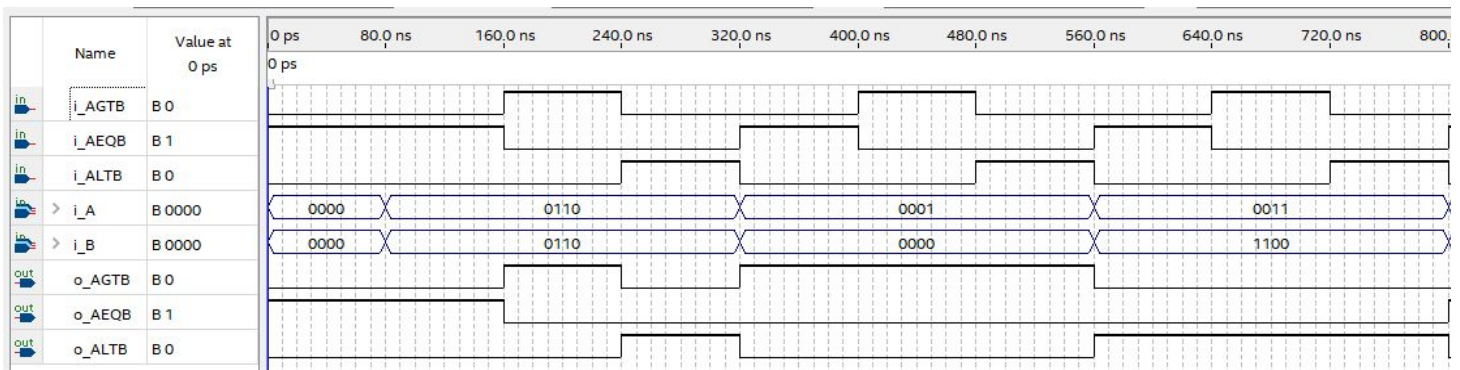
### 3) Familiarização com o Comparador de Magnitude

#### 3.1) Saídas esperadas para o componente “comparador\_85”

Teste	Sinais de Entradas	Saídas esperadas
condições iniciais	A>Bin=0, A=Bin=1, A<Bin=0, A=0000, B=0000	A>Bout=0, A=Bin=1, A<Bout=0

1	A>Bin=0, A=Bin=1, A<Bin=0, A=0110, B=0110	A>Bout=0, A=Bin=1, A<Bout=0
2	A>Bin=1, A=Bin=0, A<Bin=0, A=0110, B=0110	A>Bout=1, A=Bin=0, A<Bout=0
3	A>Bin=0, A=Bin=0, A<Bin=1, A=0110, B=0110	A>Bout=0, A=Bin=0, A<Bout=1
4	A>Bin=0, A=Bin=1, A<Bin=0, A=0001, B=0000	A>Bout=1, A=Bin=0, A<Bout=0
5	A>Bin=1, A=Bin=0, A<Bin=0, A=0001, B=0000	A>Bout=1, A=Bin=0, A<Bout=0
6	A>Bin=0, A=Bin=0, A<Bin=1, A=0001, B=0000	A>Bout=1, A=Bin=0, A<Bout=0
7	A>Bin=0, A=Bin=1, A<Bin=0, A=0011, B=1100	A>Bout=0, A=Bin=0, A<Bout=1
8	A>Bin=1, A=Bin=0, A<Bin=0, A=0011, B=1100	A>Bout=0, A=Bin=0, A<Bout=1
9	A>Bin=0, A=Bin=0, A<Bin=1, A=0011, B=1100	A>Bout=0, A=Bin=0, A<Bout=1

### 3.2) Simulação do circuito



### 3.3) Análise dos resultados

Os resultados obtidos são compatíveis com aqueles estimados no item 3.1).

### 3.4) Resumo do funcionamento do circuito

O componente “comparador\_85” tem como função comparar a magnitude de 2 números A e B, ambos possuindo 4 bits, além de oferecer a possibilidade de cascadeamento de vários comparadores iguais.

O circuito possui como entradas 4 binários que compõem A, 4 binários que compõem B e 3 sinais para cascadeamento (i\_A>B, i\_A=B, i\_A<B) para

expandir a operação para mais de 4 bits. Como saídas, temos os sinais  $o\_A>B$ , ativo quando a magnitude de A é maior que a de B,  $o\_A=B$  e  $o\_A<B$ , que funcionam de forma análoga. Para cascatear um ou mais comparadores, deve-se ligar, respectivamente, as saídas  $o\_A>B$ ,  $o\_A=B$  e  $o\_A<B$  do comparador que cuida dos dígitos mais significativos às entradas  $i\_A>B$ ,  $i\_A=B$  e  $i\_A<B$  do comparador que cuida dos dígitos menos significativos.

## 4) Projeto de um Fluxo de Dados

### 4.1) Descrição VHDL do circuito “circuito\_exp2”

#### --Descrição dos outros componentes para elaboração

-----  
-- 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 cascadeamento

```

    signal agtb : std_logic;
    signal aeqb : std_logic;
    signal altb : std_logic;
begin
    -- equacoes dos sinais: pagina 462, capitulo 6 do livro-texto
    -- Wakerly, J.F. Digital Design - Principles and Practice, 4th Edition
    -- veja tambem datasheet do CI SN7485 (Function Table)
    agtb <= (i_A3 and not(i_B3)) or
            (not(i_A3 xor i_B3) and i_A2 and not(i_B2)) or
            (not(i_A3 xor i_B3) and not(i_A2 xor i_B2) and i_A1 and not(i_B1)) or
            (not(i_A3 xor i_B3) and not(i_A2 xor i_B2) and not(i_A1 xor i_B1) and
i_A0 and not(i_B0));
    -- checa se a > b
    aeqb <= not((i_A3 xor i_B3) or (i_A2 xor i_B2) or (i_A1 xor i_B1) or (i_A0 xor
i_B0));
    -- checa se a = b
    altb <= not(agtb or aeqb);
    -- checa se a < b
    o_AGTB <= agtb or (aeqb and (not(i_AEQB) and not(i_ALTB)));
    o_ALTB <= altb or (aeqb and (not(i_AEQB) and not(i_AGTB)));
    o_AEQB <= aeqb and i_AEQB;
    -- nas saidas, são levadas em consideracao as entradas de cascadeamento
    para obter um resultado final

```

end architecture dataflow; -- fim da arquitetura

### **--descrição do fluxo de dados**

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

entity circuito_exp2 is
  port (
    clock : in std_logic;
    reset : in std_logic;
    enable : in std_logic;
    chaves : in std_logic_vector (3 downto 0);
    igual : out std_logic;
    db_contagem : out std_logic_vector (3 downto 0)
  );
end entity circuito_exp2;

architecture estrutural of circuito_exp2 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;
signal rco_out,less, great,enable_in: std_logic;
signal contador_out : std_logic_vector (3 downto 0);
begin
  contador: contador_163 port map (
    clock => clock,
    clr => reset,
    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 =>contador_out(3),
    i_B3 => chaves(3),
    i_A2 =>contador_out(2),
    i_B2 => chaves(2),
    i_A1 =>contador_out(1),
    i_B1 => chaves(1),
    i_A0 =>contador_out(0),
    i_B0 => chaves(0),
    i_AGTB =>'0',
    i_ALTB => '0',
    i_AEQB => '1',
    -- saidas
    o_AGTB =>less,
    o_ALTB =>great,

```

```
        o_AEQB => igual
    );
end architecture;
```

#### **4.2) Plano de testes para o circuito**

Caso inicial:

Entradas: clock=0,reset=1, enable=0, chaves =0,  
Aguardar uma subida de clock  
Saídas esperadas: db\_contagem=0, igual=0

Caso 1:

Entradas: clock=0,reset=1, enable=1, chaves =3,  
Aguardar 3 subidas de clock  
Saídas esperadas: db\_contagem=3, igual=1

Caso 2:

Entradas: clock=1,reset=1, enable=0, chaves =3,  
Aguardar 3 subidas de clock  
Saídas esperadas: db\_contagem=3, igual=1

Caso 3:

Entradas: clock=1,reset=1, enable=1, chaves =3,  
Aguardar 2 subidas de clock  
Saídas esperadas: db\_contagem=5, igual=0

Caso 4:

Entradas: clock=0,reset=0, enable=1, chaves =0,  
Aguardar 1 subida de clock  
Saídas esperadas: db\_contagem=0, igual=1

Caso 5:

Entradas: clock=1,reset=1, enable=1, chaves =10,  
Aguardar 4 subidas de clock  
Saídas esperadas: db\_contagem=4, igual=0

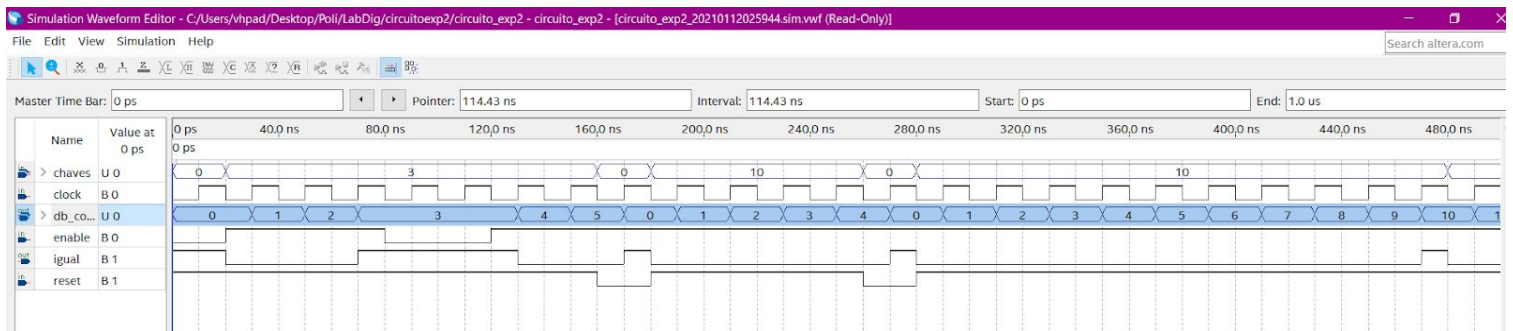
Caso 6:

Entradas: clock=0,reset=0, enable=1, chaves =0,  
Aguardar 1 subidas de clock  
Saídas esperadas: db\_contagem=0, igual=1

Caso 7:

Entradas: clock=0,reset=1, enable=1, chaves =10,  
Aguardar 10 subidas de clock  
Saídas esperadas: db\_contagem=10, igual=1

#### **4.3) Simulação do circuito**



#### 4.4) Descrição do comportamento do circuito

O circuito faz uma contagem até um valor desejado. Esse valor é dado pela entrada “chaves”, levando a saída “igual” para um nível lógico alto quando a contagem chegar ao valor desejado. O valor atual da contagem é dado pela saída db\_contagem. A entrada enable habilita o componente e a entrada reset reinicia a contagem.