Projeto de Circuitos Digitais em VHDL

Experimento #5

PCS3335 - Laboratório Digital A

26/05/2022

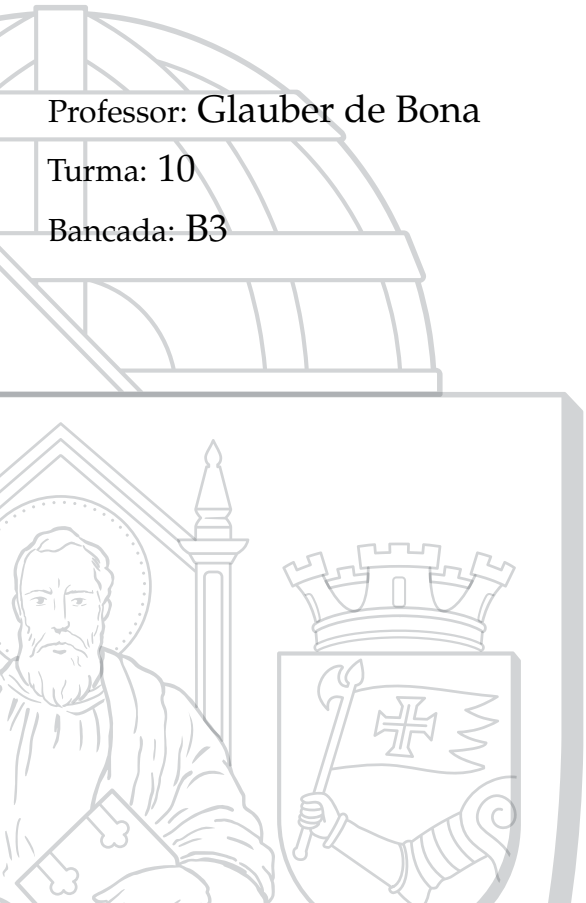
Natanael Magalhães Cardoso, 8914122

Renato Naves Fleury, 11805269

Professor: Glauber de Bona

Turma: 10

Bancada: B3



Universidade de São Paulo

Escola Politécnica

Departamento de Eng. de Computação e Sistemas Digitais

UNIVERSIDADE DE SÃO PAULO

ESCOLA POLITÉCNICA

Departamento de Eng. de Computação e Sistemas Digitais

Projeto de Circuitos Digitais em VHDL

Natanael Magalhães Cardoso, Renato Naves Fleury

1. INTRODUÇÃO

O VHDL é uma linguagem de descrição de hardware. Com ela é possível descrever e simular circuitos digitais dos mais diversos sem a necessidade de se montar o circuito e testá-lo a cada alteração na descrição. Nesse experimento, será realizado um projeto simples utilizadno o VHDL.

2. OBJETIVOS

O objetivo deste experimento é desenvolver o projeto de um comparador, um contador direcional e outro bidirecional usando VHDL sob o paradígma comportamental e usar os componentes menores para desenvolver um projeto de controle de vagas de estacionamento usando VHDL sob o paradígma estrutural.

3. PLANEJAMENTO

3.1. COMPARADOR E CONTADOR EM VHDL

3.1.1 Implementação

Os cirucitos forma implementados em VHDL usando o paradigma comportamental. A descrição comentada do circuito comparador está disposta na Listagem 4 e a descrição do circuito contador está na Listagem 5 no Apêndice A.

3.1.2 Simulação

As cartas dos tempos do circuito comparador e comparador estão dispostas nas Figs. 1 e 2, respectivamente.

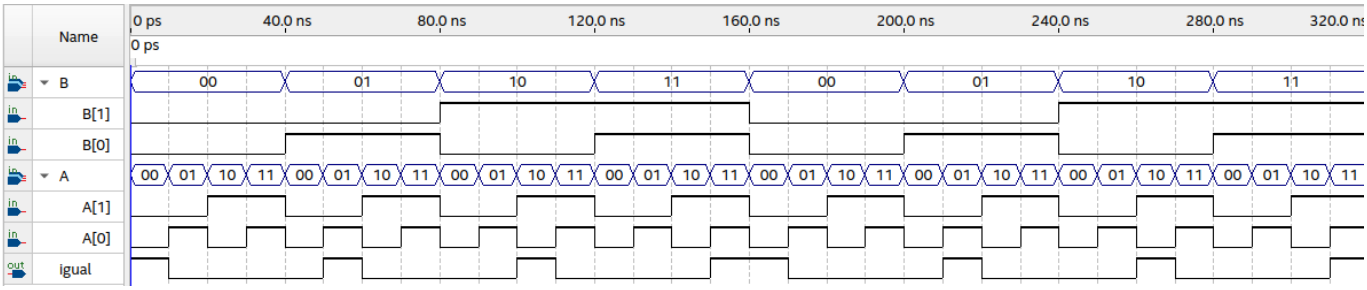


Figura 1: Carta dos tempos para o circuito comparador

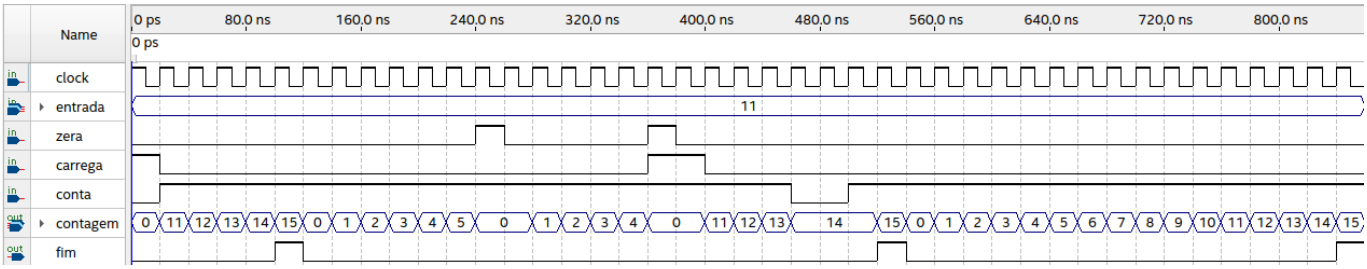


Figura 2: Carta dos tempos para o circuito contador

3.1.3 Contador Bidirecional

Foi adicionado um novo sinal de entrada denominado "direção", que controla o comportamento da contagem. A contagem é crescente caso esse sinal seja alto e descrescente caso contrário. Com essa modificação foi implementado um circuito contador bidirecional UP/$\overline{\text{DOWN}}$. A Listagem 6 mostra a descrição deste circuito e a Fig. 3 mostra a carta dos tempos.

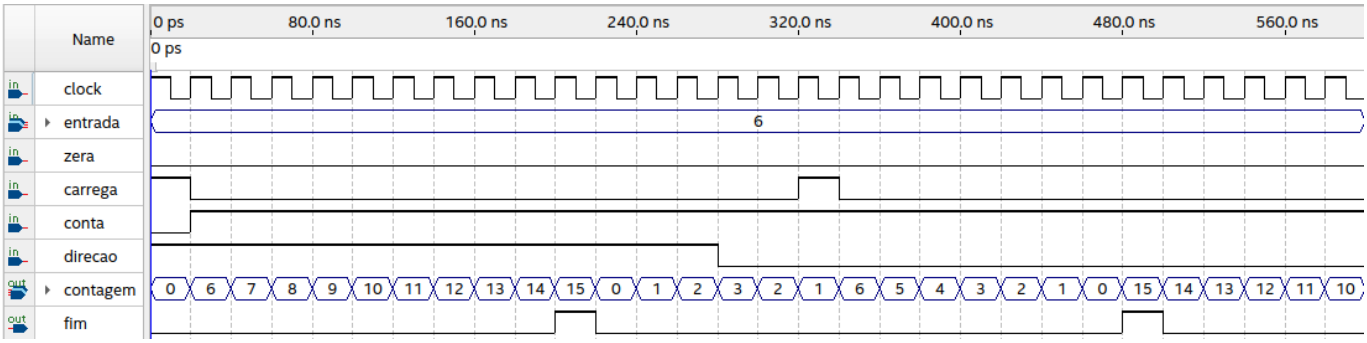


Figura 3: Carta dos tempos do circuito contador bidirecional

3.2. Controle de Vagas de Estacionamento

3.2.1 Implementação

De modo a atender a necessidade de um contagem acíclica e bidirecional para o projeto do controle de vagas de estacionamento, a descrição do contador bidirecional (6) foi alterada de modo a limitar a contagem de 0 a 15 de forma acíclica. Essa nova descrição se encontra na Listagem 1.

Listing 1: Descrição do circuito contador bidirecional acíclico de 4 bits.

```
-- contador.vhd
-- contador bidirecional acíclico de 4 bits

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity contador is
  port (
    clock, zera, conta, direcao: in std_logic;
    contagem: out std_logic_vector(3 downto 0)
  );
end contador;

architecture contador_arch of contador is
  signal IQ: integer range 0 to 15;
begin
  process(clock, zera, conta)
  begin
    if zera='1' then
      IQ <= 0;
    elsif clock'event and clock='1' then
      if conta='1'then
                        if (direcao='1' and IQ<15) then
                                IQ <= IQ + 1;
                        elsif (direcao='0' and IQ>0) then
                                IQ <= IQ - 1;
                        end if;
                else
                        IQ <= IQ;
                end if;
        end if;
  end process;
```

```
34    contagem <= std_logic_vector(to_unsigned(IQ, contagem'length));
35  end contador_arch;
```

Além disso também foram implementadas a descrição do sinalizador de vagas e a descrição estrutural da integração desses dois circuitos, ambos podem ser vistos nas Listagens 2 e 3.

Listing 2: Descrição do circuito Sinalizador de Vagas.

```
1   -- sinalizador.vhd
2   -- sinalizador de vagas
3
4   library IEEE;
5   use IEEE.std_logic_1164.all;
6   use IEEE.numeric_std.all;
7
8   entity sinalizador is
9     port (
10      vagas, contagem: in std_logic_vector(3 downto 0);
11      fim: out std_logic
12    );
13  end sinalizador;
14
15  architecture sinalizador_arch of sinalizador is
16    begin
17
18          fim <= '0' when vagas>contagem else '1';
19
20  end sinalizador_arch;
```

Listing 3: Descrição estrutural de um Controle de Vagas de Estacionamento.

```
1   -- controle.vhd
2   -- controle de vagas de estacionamento
3
4   library IEEE;
5   use IEEE.std_logic_1164.all;
6   use IEEE.numeric_std.all;
7
8   entity controle is
9     port (
10      clock, zera, conta, direcao: in std_logic;
11           vagas: in std_logic_vector(3 downto 0);
```

```
12          contagem: out std_logic_vector(3 downto 0);
13          cheio: out std_logic
14    );
15  end controle;
16
17  architecture controle_arch of controle is
18
19          component contador is
20            port (
21                  clock, zera, conta, direcao: in std_logic;
22                  contagem: out std_logic_vector(3 downto 0)
23            );
24          end component;
25
26          component sinalizador is
27            port (
28                  vagas, contagem: in std_logic_vector(3 downto 0);
29                  fim: out std_logic
30            );
31          end component;
32
33          signal quant_vagas: std_logic_vector(3 downto 0);
34
35  begin
36          contagem <= quant_vagas;
37          contador_1: contador port map (clock, zera, conta, direcao, quant_vagas);
38          sinalizador_1: sinalizador port map (vagas, quant_vagas, cheio);
39  end controle_arch;
```

3.2.2 Diagrama de blocos

O diagrama de blocos obtido com a função RTL Viewer da ferramenta Quartus na Figura 4.

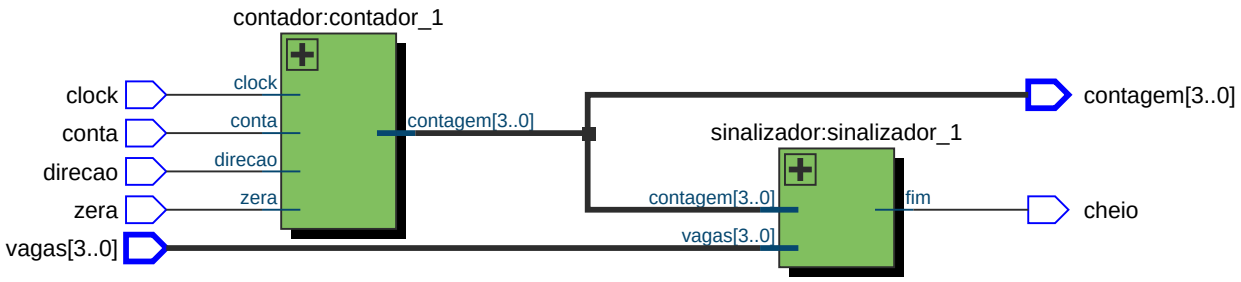


Figura 4: Diagrama de blocos do Controle de Vagas de Estacionamento

O sinal de saída "contagem" foi utilizado apenas para facilitar a depuração.

3.2.3 Simulação

A carta de tempos que simula o funcionamento do circuito pode ser vista na Figura 5. Para essa simulação tentou-se imitar uma situação na qual o número máximo de vagas é 6 e entram carros até encher o estacionamento e depois esses carros saem até esvaziá-lo.

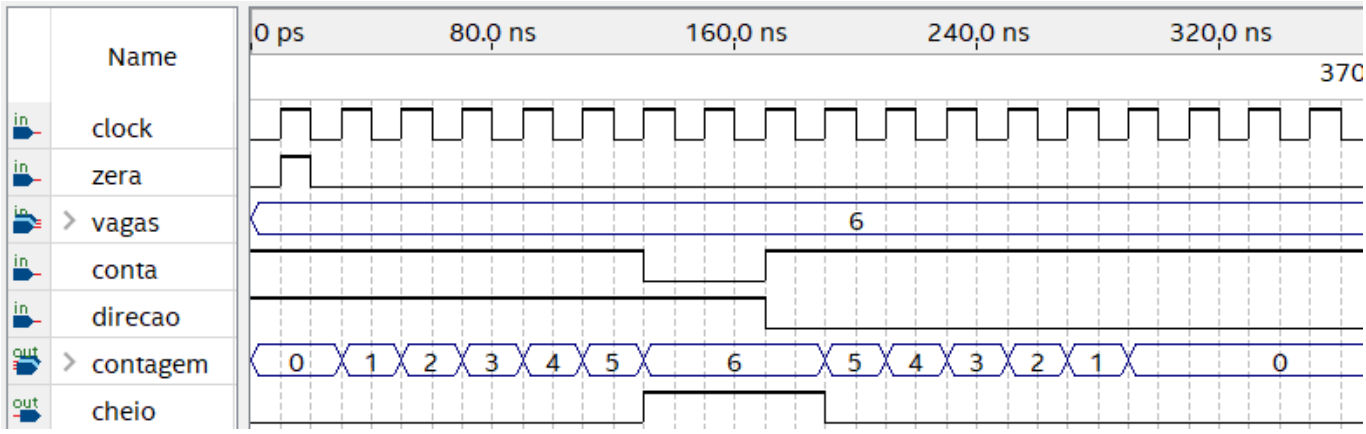


Figura 5: Carta de tempos da simulação do circuito Controle de Vagas de Estacionamento.

3.2.4 Tabela de testes

A Tabela 1 motra a Tabela de Testes do circuito de controle de vagas

Tabela 1: Tabela de testes do circuito de controle de vagas

Entradas					Depuração	Saída
clk	zera	vagas	conta	direcao	contagem	cheio
×	1	×	×	×	000	0
↑	0	110	1	1	001	0
↑	0	110	1	1	010	0
↑	0	110	1	1	011	0
↑	0	110	1	1	100	0
↑	0	110	1	1	101	0
↑	0	110	1	1	110	1
↑	0	110	1	0	110	1
↑	0	110	1	0	101	0
↑	0	110	1	0	100	0
↑	0	110	1	0	011	0
↑	0	110	1	0	010	0
↑	0	110	1	0	001	0
↑	0	110	1	0	000	0

4. Resultados

O sistema digital planejado foi implantado na placa FPGA com sucesso. O circuito apresentou o comportamento desejado para todos os testes e nenhuma alteração do projeto foi necessária. Adicionalmente, foi feita uma modificação do projeto para que o contador parasse a contagem assim que atingisse o valor máximo de vagas.

5. Desafio

Foi incluído um novo sinal de entrada no circuito contador de vagas de estacionamento que, quando em nível lógico alto, modifica o contador para incrementar/decrementar de duas unidades, ao invés de uma, e, quando em nível lógico baixo, não modifica nenhum comportamento do circuito, ou seja, continua contado de um em um.

As Listagens 8 e 7 mostram as diferenças entre as descrições anteriores (1 e 3) e as atuais para os componentes `contador` e `controle`, respectivamente, e o componente `sinalizador` não foi alterado. A Fig. 6 mostra o diagrama de blocos do novo circuito, alterações incluem o recebimento do sinal `duplo` e a realimentação do sinal `cheio` no contador.

Após a montagem, o circuito apresentou todos os comportamentos esperados.

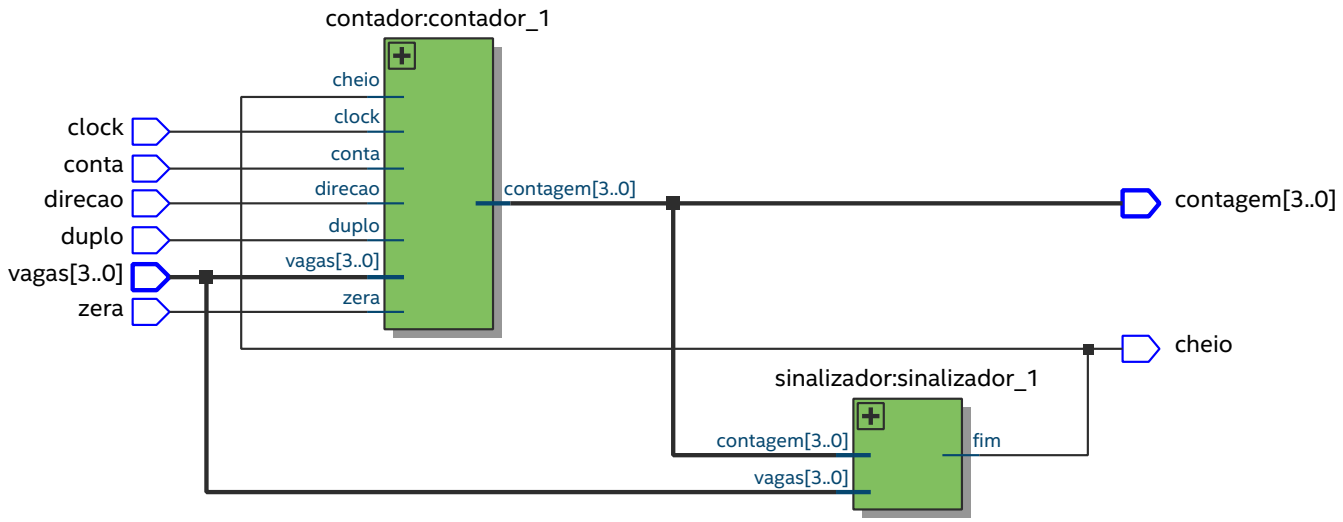


Figura 6: Diagrama de blocos do circuito considerando contagem dupla

6. Conclusão

Com este experimento, pudemos projetar, simular e implantar um circuito digital para controle de vagas em um estacionamento em uma placa FPGA. A descrição deste circuito foi feita explorando dois paradígmas do VHDL: o comportamental, usado para descrever os componentes menores, e o estrutural, usado para unir os componentes do circuito. Durante a implantação não houve nenhuma alteração do planejamento e o circuito operou como planejado.

APÊNDICE

A. DESCRIÇÕES DOS CIRCUITOS

Descrições comentadas dos circuitos comparador e contador em VHDL

Listing 4: Descrição do circuito comparador

```
1  -- comparador.vhd
2  -- comparador binario com entradas de 2 bits
3
4  library IEEE;
5  use IEEE.std_logic_1164.all;
6
7  -- definição das portas de entrada e saída
8  entity comparador is
9    port (
10     A, B: in std_logic_vector(1 downto 0);
11     igual: out std_logic
12   );
13 end comparador;
14
15 -- descrição das ligações internas do circuito
16 architecture comportamental of comparador is
17 begin
18   -- o sinal igual terá nível lógico alto sempre que os sinais A e B
19   -- tiverem o mesmo nível lógico, caso contrário terá nível lógico baixo
20   igual <= '1' when A=B else '0';
21 end comportamental;
```

Listing 5: Descrição do circuito contador

```
-- contador.vhd
-- contador hexadecimal de 4 bits

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

-- definição dos sinais de entrada e saída do circuito
entity contador is
  port (
    clock, zera, conta, carrega: in std_logic;
    entrada: in std_logic_vector(3 downto 0);
    contagem: out std_logic_vector(3 downto 0);
    fim: out std_logic
  );
end contador;


-- implementação comportamental do circuito contador
architecture comportamental of contador is
  -- sinal IQ: sinal do tipo inteiro que armazena o valor atual do contador
  signal IQ: integer range 0 to 15;
begin
  process(clock, zera, conta, carrega, entrada, IQ)
  -- define um processo sensível à mudança dos níveis lógicos dos sinais
  -- clock, zera, conta, carrega, entrada e IQ
  -- os sinais zera, conta e carrega definem as ações do circuito
  begin
    if zera='1' then
      -- clear assíncrono (não depende da borda de subida do clock)
      IQ <= 0; -- zera o valor do contador

    elsif clock'event and clock='1' then
      -- ações sensíveis à borda de subida do clock são colocadas aqui
      -- há duas ações possíveis: carregar os dados (maior precedência)
      -- e incrementar o valor do sinal em 1 (menor precedência)
      -- apenas uma das ações é performada por clock

      if carrega='1' then
```

```
40          -- ação "carrega": configura o contador para receber
41          -- um valor específico de um sinal de entrada do circuito
42          IQ <= to_integer(unsigned(entrada));
43
44        elsif conta='1' then
45          -- ação "conta": lógica de um acumulador circular,
46          -- incrementa o valor atual em 1 se o valor atual for menor
47          -- que o valor máximo, zera caso contrário
48          if IQ=15 then
49            IQ <= 0;
50          else
51            IQ <= IQ + 1;
52          end if;
53
54        else
55          -- se nenhuma ação for especificada, o contador permanece com
56          -- o sinal atual
57          IQ <= IQ;
58        end if;
59      end if;
60    end process;
61
62    -- liga do sinal interno IQ ao sinal de saída do contador
63    -- faz cast de inteiro para std_logic_vector
64    contagem <= std_logic_vector(to_unsigned(IQ, contagem'length));
65
66    -- detecta quando o contador atinge o valor máximo
67    fim <= '1' when IQ=15 else '0';
68  end comportamental;
```

Listing 6: Descrição do circuito contador UP/$\overline{\text{DOWN}}$. Os comentários no código foram omitidos, já que as modificações estão comentadas na Seção 3.1.3

```
1  -- contador_ud.vhd
2  -- contador hexadecimal bidirecional de 4 bits
3
4  library IEEE;
5  use IEEE.std_logic_1164.all;
6  use IEEE.numeric_std.all;
7
8
9  entity contador_ud is
10   port (
11     clock, zera, conta, carrega, direcao: in std_logic;
12     entrada: in std_logic_vector(3 downto 0);
13     contagem: out std_logic_vector(3 downto 0);
14     fim: out std_logic
15   );
16 end contador_ud;
17
18
19 architecture comportamental of contador_ud is
20   signal IQ: integer range 0 to 15;
21 begin
22   process(clock, zera, conta, carrega, direcao, entrada, IQ)
23   begin
24     if zera='1' then
25       IQ <= 0;
26     elsif clock'event and clock='1' then
27       if carrega='1' then
28       IQ <= to_integer(unsigned(entrada));
29       elsif conta='1' then
30         if direcao='1' then
31           if IQ=15 then
32             IQ <= 0;
33           else
34             IQ <= IQ + 1;
35           end if;
36         else
37           if IQ=0 then
38             IQ <= 15;
```

```
39            else
40              IQ <= IQ - 1;
41            end if;
42          end if;
43        else
44          IQ <= IQ;
45        end if;
46      end if;
47    end process;
48
49    contagem <= std_logic_vector(to_unsigned(IQ, contagem'length));
50    fim <= '1' when IQ=15 else '0';
51  end comportamental;
```

Listing 7: Alterações na descrição do componente `controle.vhd` para implementação da contagem dupla[1]

```
@@ -7,7 +7,7 @@

 entity controle is
   port (
-    clock, zera, conta, direcao: in std_logic;
+    clock, zera, conta, direcao, duplo: in std_logic;
     vagas: in std_logic_vector(3 downto 0);
     contagem: out std_logic_vector(3 downto 0);
     cheio: out std_logic
@@ -18,7 +18,8 @@

   component contador is
     port (
-      clock, zera, conta, direcao: in std_logic;
+      clock, zera, conta, direcao, cheio, duplo: in std_logic;
+      vagas: in std_logic_vector(3 downto 0);
       contagem: out std_logic_vector(3 downto 0)
     );
   end component;
@@ -31,9 +32,12 @@
   end component;

   signal quant_vagas: std_logic_vector(3 downto 0);
+  signal re_conta, re_cheio: std_logic;

 begin
   contagem <= quant_vagas;
-  contador_1: contador port map (clock, zera, conta, direcao, quant_vagas);
-  sinalizador_1: sinalizador port map (vagas, quant_vagas, cheio);
+  cheio <= re_cheio;
+  contador_1: contador port map (clock, zera, conta, direcao, re_cheio,
+    duplo, vagas, quant_vagas);
+  sinalizador_1: sinalizador port map (vagas, quant_vagas, re_cheio);
 end controle_arch;
```

[1] Gerado com `diff -Naru controle_vagas/controle.vhd desafio/controle.vhd`

Listing 8: Alterações na descrição do componente `contador.vhd` para implementação da contagem dupla[2]

```
@@ -8,25 +8,35 @@

 entity contador is
   port (
-    clock, zera, conta, direcao: in std_logic;
+    clock, zera, conta, direcao, cheio, duplo: in std_logic;
+    vagas: in std_logic_vector(3 downto 0);
     contagem: out std_logic_vector(3 downto 0)
   );
 end contador;


 architecture contador_arch of contador is
-  signal IQ: integer range 0 to 15;
+  signal IQ, int_vagas: integer range 0 to 15;
 begin
+  int_vagas <= to_integer(unsigned(vagas));
   process(clock, zera, conta)
   begin
     if zera='1' then
       IQ <= 0;
     elsif clock'event and clock='1' then
       if conta='1'then
-        if (direcao='1' and IQ<15) then
-          IQ <= IQ + 1;
-        elsif (direcao='0' and IQ>0) then
-          IQ <= IQ - 1;
+        if duplo='0' then
+          if (direcao='1' and IQ<15 and cheio='0') then
+            IQ <= IQ + 1;
+          elsif (direcao='0' and IQ>0) then
+            IQ <= IQ - 1;
+          end if;
+        else
+          if (direcao='1' and IQ<(int_vagas-1) and cheio='0') then
+            IQ <= IQ + 2;
+          elsif (direcao='0' and IQ>1) then
+            IQ <= IQ - 2;
```

```
39 +           end if;
40           end if;
41         else
42           IQ <= IQ;
```

[2]Gerado com `diff -Naru controle_vagas/contador.vhd desafio/contador.vhd`

Universidade de São Paulo

Escola Politécnica

Departamento de Eng. de Computação e Sistemas Digitais