Projeto de Máquinas de Estados em VHDL

Experimento #6

PCS3335 - Laboratório Digital A

02/06/2022

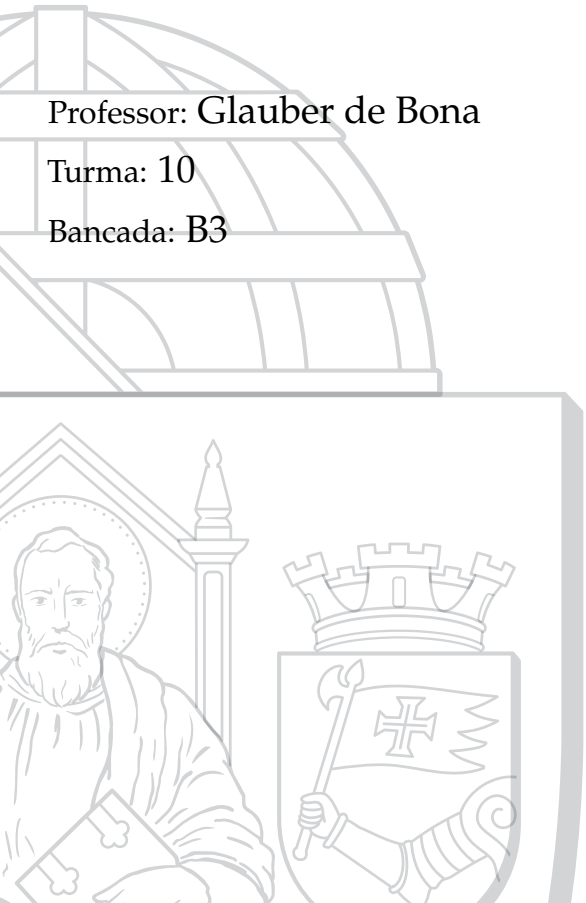
Natanael Magalhães Cardoso, 8914122

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 Máquinas de Estados 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 modificar o projeto anterior de controle de vagas de estacionamento adicionando contagem de veículos pertencentes a idosos e implementá-lo usando a divisão em fluxo de dados e unidade de controle.

3. PLANEJAMENTO

3.1. FLUXO DE DADOS

3.1.1 Diagrama de blocos

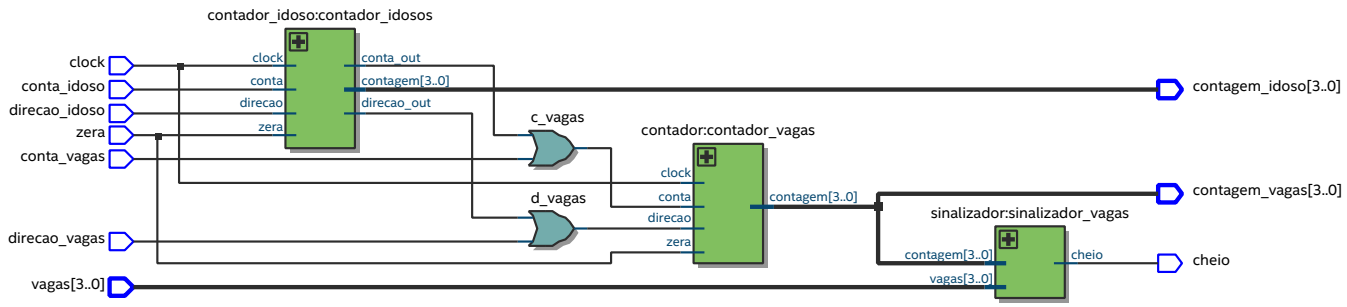


Figura 1: Diagrama de blocos do Fluxo de Dados

A Fig. 1 mostra o Diagrama de Blocos gerado a partir da descrição em VHDL do fluxos de dados (1).

3.1.2 Carta dos Tempos

A partir da simulação, foi possível gerar a carta de tempos para o componente fluxo de dados, que é mostrada na Fig. 2. Nela é testada se o contador dá prioridade ao contador idoso. Também é notado que o contador não para no limite de vagas, pois este comportamento é controlado pela unidade de controle.

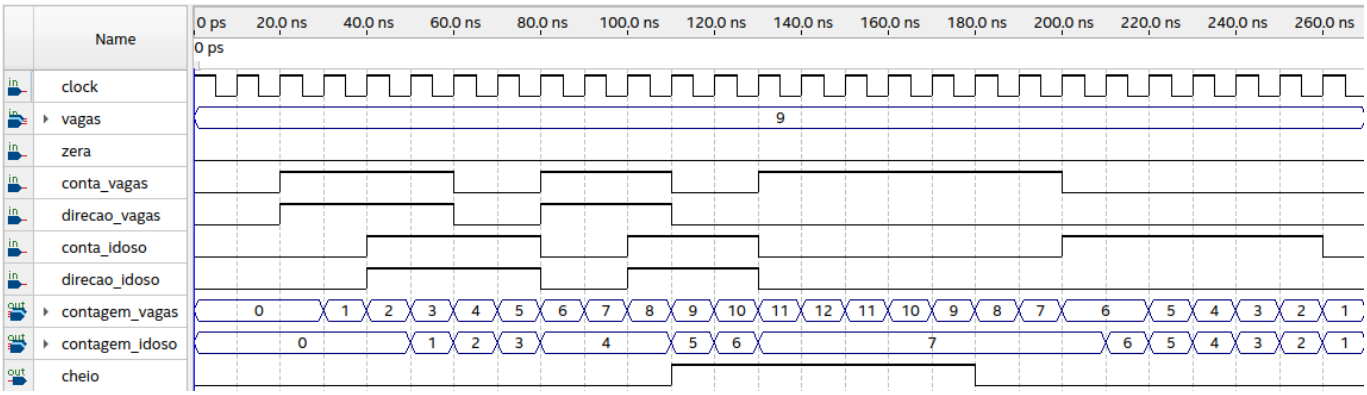


Figura 2: Carta de tempos do Fluxo de Dados

3.2. Unidade de Controle

3.2.1 Diagrama de blocos

A Fig. 3 mostra o Diagrama de Blocos gerado a partir da descrição em VHDL do fluxos de dados (2).

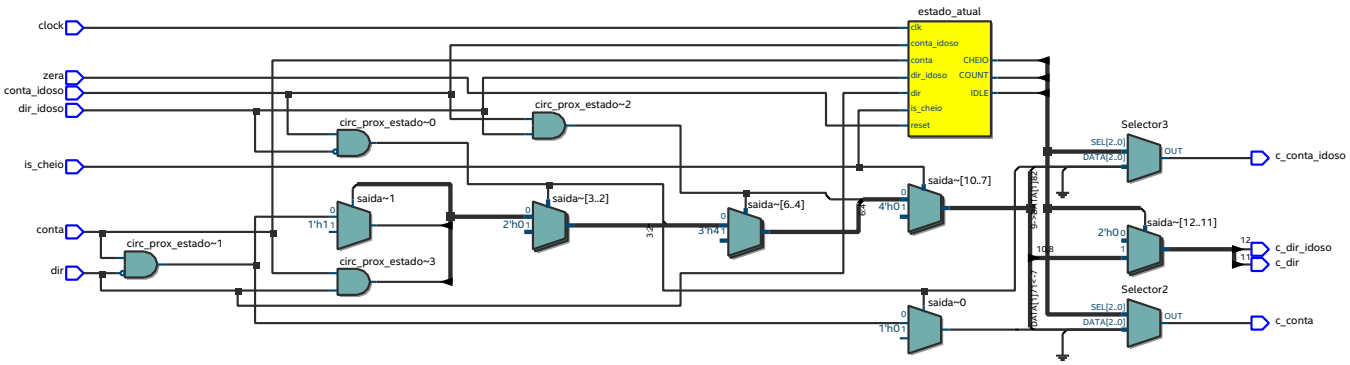


Figura 3: Diagrama de blocos da Unidade de Controle

3.2.2 Diagrama de estados

A Fig. 4 mostra o diagrama de estados para a Máquina de Estados implementada na Unidade de Controle. Ela possui dois estados: "CHEIO" e "COUNT", sendo que o terceiro estado foi colocado apenas por restrições da ferramenta Quartus.

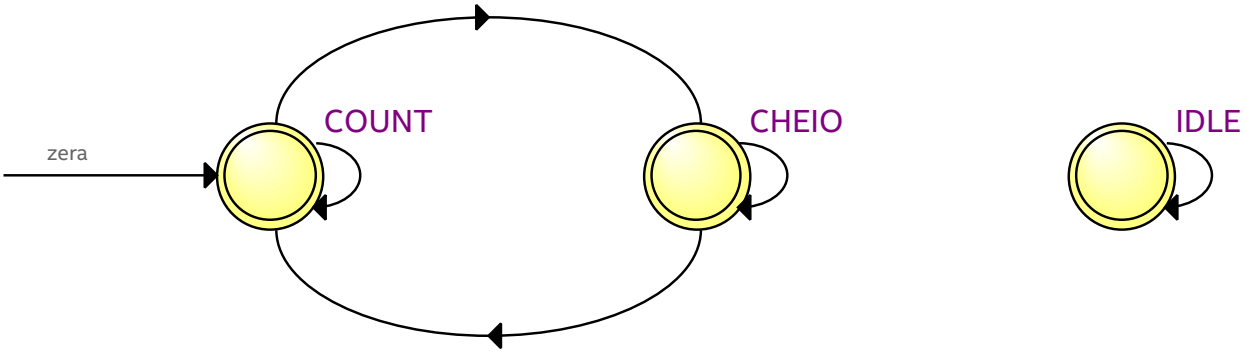


Figura 4: Diagrama de estados da Unidade de Controle

3.2.3 Carta dos Tempos

A partir da simulação, foi possível gerar a carta de tempos para o componente unidade de controle, que é mostrada na Fig. 5. Nela são testados os sinais de controle gerados a partir dos estímulos externos para controlar o componente fluxo de dados.

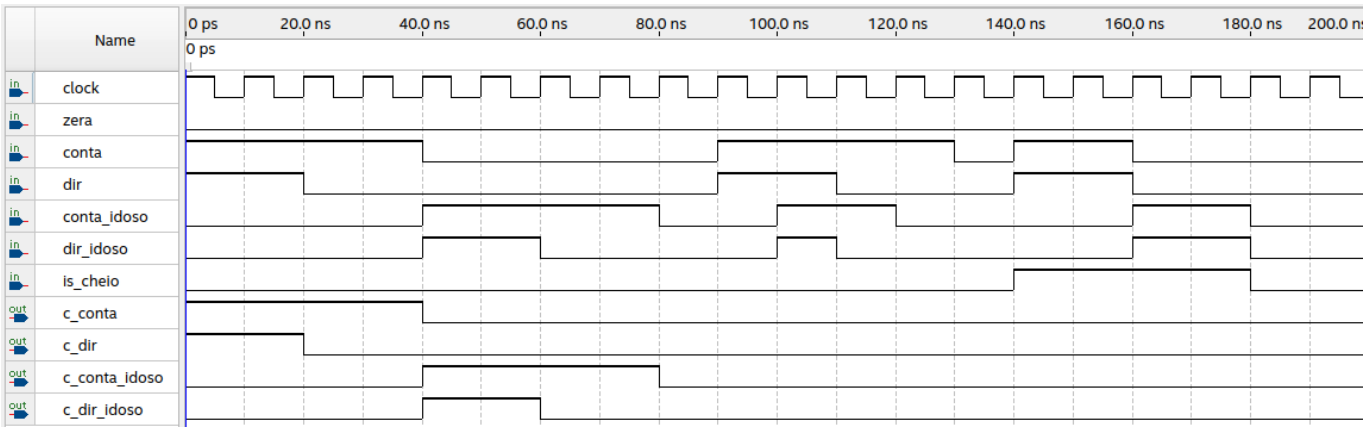


Figura 5: Carta de tempos da Unidade de Controle

4. RESULTADOS

Foi implementada descrição em VHDL do circuito completo, que inclui os componentes *Fluxo de Dados* e *Unidade de Controle*. A Fig. 6 mostra a carta de tempos deste circuito completo.

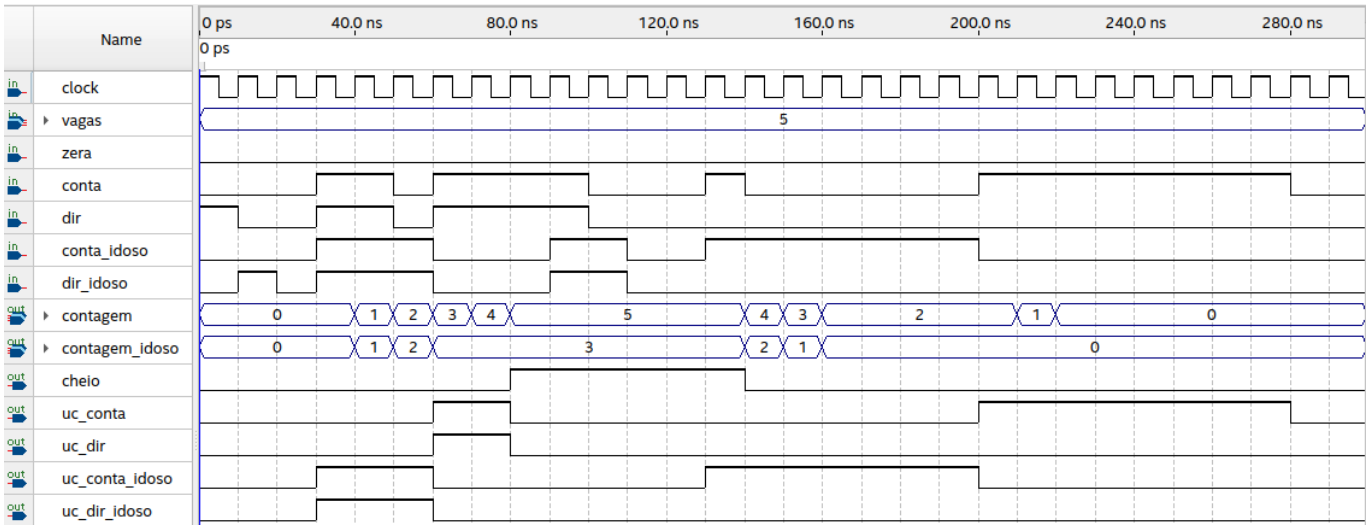


Figura 6: Carta de tempos do circuito completo

Após a implementação da descrição, o circuito foi implantado em uma placa FPGA. A Tabela 1 mostra os valores esperados para cada teste. Todas as respostas do circuito implantado foram condizentes com a tabela de testes.

Tabela 1: Tabela de testes do circuito completo

Entradas							Saídas		
clk	zera	vagas	conta	direcao	conta (i)	direcao (i)	contagem	contagem (i)	cheio
×	1	×	×	×	×	×	000	000	0
↑	0	110	1	1	0	0	001	000	0
↑	0	110	1	1	0	0	010	000	0
↑	0	110	1	1	0	0	011	000	0
↑	0	110	0	0	1	1	100	001	0
↑	0	110	0	0	1	1	101	010	0
↑	0	110	1	1	0	0	110	010	1
↑	0	110	1	0	0	0	110	010	1
↑	0	110	1	0	0	0	101	000	0
↑	0	110	1	0	0	0	100	000	0
↑	0	110	1	0	0	0	011	000	0
↑	0	110	1	0	0	0	010	000	0
↑	0	110	0	0	1	0	001	001	0
↑	0	110	0	0	1	0	000	000	0

5. CONCLUSÃO

Um circuito digital para controle de vagas de estacionamento com contagem separada de idosos foi implementado em VHDL e implantado em uma placa FPGA com todas as saídas desse circuito coerentes com os resultados de simulação.

APÊNDICE

A. DESCRIÇÕES DOS CIRCUITOS

Descrições comentadas dos circuitos comparador e contador em VHDL

Listing 1: Descrição do Fluxo de Dados

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  entity fluxo_dados is
6    port(
7      clock, zera: in std_logic;
8      vagas: in std_logic_vector(3 downto 0);
9      conta_vagas, direcao_vagas: in std_logic;
10     conta_idoso, direcao_idoso: in std_logic;
11     contagem_vagas, contagem_idoso: out std_logic_vector(3 downto 0);
12     cheio: out std_logic
13   );
14 end fluxo_dados;
15
16 architecture fluxo_dados_arch of fluxo_dados is
17   component contador is
18     port(
19       clock, zera, conta, direcao: in std_logic;
20       contagem: out std_logic_vector(3 downto 0)
21     );
22   end component;
23
24   component contador_idoso is
25     port(
26       clock, zera, conta, direcao: in std_logic;
27       contagem: out std_logic_vector(3 downto 0);
28       conta_out, direcao_out: out std_logic
29     );
30   end component;
31
32   component sinalizador is
33     port(
```

```
      vagas, contagem: in std_logic_vector(3 downto 0);
      cheio: out std_logic
    );
  end component;

  signal qtd_vagas, qtd_idoso: std_logic_vector(3 downto 0);
  signal c_idoso, d_idoso, c_vagas, d_vagas: std_logic;
begin
  contagem_vagas <= qtd_vagas;
  contagem_idoso <= qtd_idoso;
  c_vagas <= conta_vagas or c_idoso;
  d_vagas <= direcao_vagas or d_idoso;

  contador_vagas: contador port map(
    clock => clock,
    zera => zera,
    conta => c_vagas,
    direcao => d_vagas,
    contagem => qtd_vagas
  );

  contador_idosos: contador_idoso port map(
    clock => clock,
    zera => zera,
    conta => conta_idoso,
    direcao => direcao_idoso,
    contagem => qtd_idoso,
    conta_out => c_idoso,
    direcao_out => d_idoso
  );

  sinalizador_vagas: sinalizador port map(
    vagas => vagas,
    contagem => qtd_vagas,
    cheio => cheio
  );
end fluxo_dados_arch;
```

Listing 2: Descrição da Unidade de Controle

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

entity unidade_controle is
  port(
    clock, zera, is_cheio: in std_logic;
    conta, dir, conta_idoso, dir_idoso: in std_logic;
    c_conta, c_dir, c_conta_idoso, c_dir_idoso: out std_logic
  );
end unidade_controle;


architecture unidade_controle_arch of unidade_controle is
  type estado_t is (COUNT, CHEIO, IDLE);
  signal estado_prox, estado_atual: estado_t;
  signal saida: std_logic_vector(3 downto 0);
begin
  sincrono: process(clock, zera, estado_prox)
  begin
    if zera='1' then
      estado_atual <= COUNT;
    elsif clock'event and clock='1' then
      estado_atual <= estado_prox;
    end if;
  end process;

  circ_prox_estado: process(estado_atual, conta, dir, conta_idoso, dir_idoso,
  ↪ is_cheio)
  begin
    case estado_atual is
      when CHEIO =>
        if conta_idoso='1' and dir_idoso='0' then
          estado_prox <= COUNT;
          saida <= "0010";
        elsif conta='1' and dir='0' then
          estado_prox <= COUNT;
          saida <= "1000";
        else
```

```
39            estado_prox <= CHEIO;
40            saida <= "0000";
41          end if;
42        when COUNT =>
43          if is_cheio='1' then
44            estado_prox <= CHEIO;
45            saida <= "0000";
46          elsif conta_idoso='1' and dir_idoso='1' then
47            estado_prox <= COUNT;
48            saida <= "0011";
49          elsif conta_idoso='1' and dir_idoso='0' then
50            estado_prox <= COUNT;
51            saida <= "0010";
52          elsif conta='1' and dir='1' then
53            estado_prox <= COUNT;
54            saida <= "1100";
55          elsif conta='1' and dir='0' then
56            estado_prox <= COUNT;
57            saida <= "1000";
58          else
59            estado_prox <= COUNT;
60            saida <= "0000";
61          end if;
62        when IDLE =>
63          estado_prox <= IDLE;
64          saida <= "0000";
65        end case;
66    end process;
67
68    c_conta <= saida(3);
69    c_dir <= saida(2);
70    c_conta_idoso <= saida(1);
71    c_dir_idoso <= saida(0);
72  end unidade_controle_arch;
```

Universidade de São Paulo

Escola Politécnica

Departamento de Eng. de Computação e Sistemas Digitais