Disciplina: PCS 3335 – Laboratório Digital A	
Prof.: *Tereza*	**Data:** *28/05*
Turma: *1*	**Bancada:** *A7*
Membros:	
9848836 Bruna Okura	
10773801 Gabriel Tavares	

Experiência *6*

Máquinas de Estados em VHDL

1. Introdução

Nesta experiência são estudadas máquinas de estados descritas em VHDL e sua aplicação em um circuito digital simples.

2. Objetivo

Esta experiência tem como objetivo aprender sobre descrição de máquinas de estados em VHDL, a aplicação dessas máquinas, como unidade de controle de um circuito digital e o estudo de um circuito digital simples. Ao final, teremos conhecimento sobre o desenvolvimento de sistemas digitais mais complexos, compostos por fluxo de dados e unidades de controle.

3. Planejamento

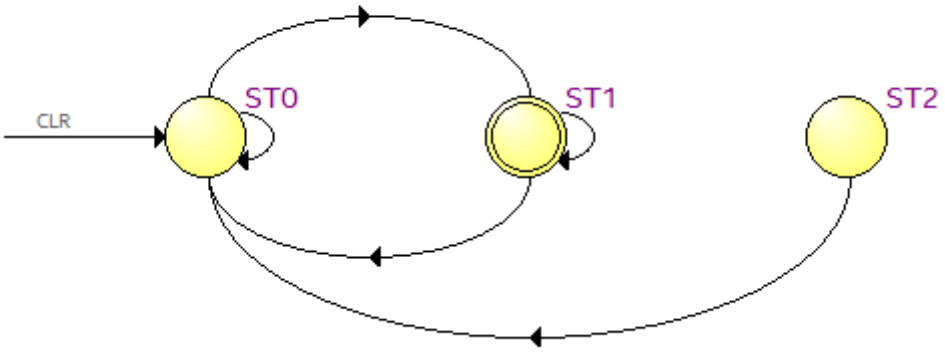
a) Projeto de uma Máquina de Estados em VHDL

i) Pseudocódigo da Descrição Comportamental

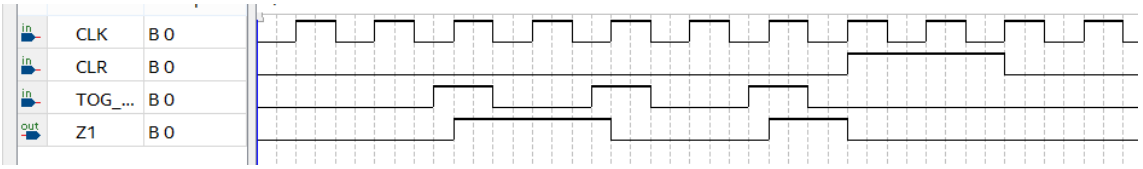
```
TOG_EN, CLK, CLRL: in
Z1: out

PS, NS : state_type (ST0, ST1)

process
    if (CLR = 1)
        {PS = ST0}
    if (rising_edge(CLK))
        {PS = NS}
end process

process
    Z1 = 0
    if (PS = ST0)
        {Z1 = 0
        if (TOG_EN = 1)
            {NS = ST1}
        else
            {NS = ST0}
        }
    if (PS = ST1)
        {Z1 = 1
        if (TOG_EN = 1)
            {NS = ST0}
        else
            {NS = ST1}
        }
    else {
        Z1 = 0
        NS = ST0
    }
end process
```

ii) Diagrama de Transição de Estados

O Diagrama de Transição obtido pelo programa é similar ao do enunciado, a única diferença é a presença de um terceiro estado (ST2). Este estado foi adicionado pois para uma máquina ser considerada uma máquina de Moore, é necessário que haja pelo menos 3 estados.



iii) Carta de Tempos



b) Projeto de um Fluxo de Dados em VHDL

i) Descrição funcional

O circuito recebe 7 entradas de 1 bit cada (clock, clear, conta_idosos, conta_normal, entra_sai_normal, entra_sai_idosos, load), 2 entradas de 4 bits cada (vagas_total, vagas_load) e retorna 2 saídasde 4 bits cada (vagas_out_total, vagas_out_idosos) e 1 saída de 1 bit (cheio).

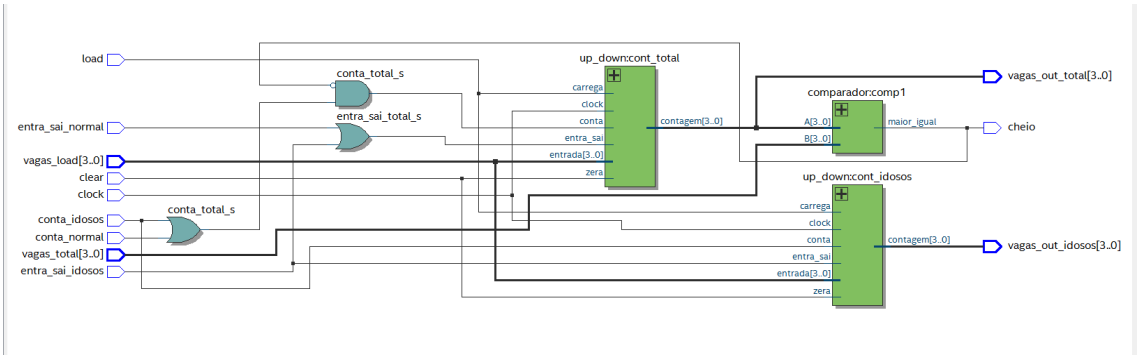
ii) Circuito em VHDL

```vhdl
-- contador e sinalizador de vagas
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.numeric_std.all;

entity controlador_vagas is
    port (
        clock, clear, conta_idosos, conta_normal, entra_sai_normal, entra_sai_idosos, load: in std_
        vagas_total, vagas_load: in std_logic_vector (3 downto 0);
        vagas_out_total, vagas_out_idosos : out std_logic_vector (3 downto 0);
        cheio: out std_logic
    );
end controlador_vagas;

architecture arch of controlador_vagas is

component comparador is
    port (
        A, B: in std_logic_vector (3 downto 0);
        maior_igual: out std_logic
    );
end component comparador;

component up_down is
    port (
        clock, zera, conta, carrega, entra_sai: in std_logic;

        entrada: in std_logic_vector (3 downto 0);
        contagem: out std_logic_vector (3 downto 0)
    );
end component up_down;
```

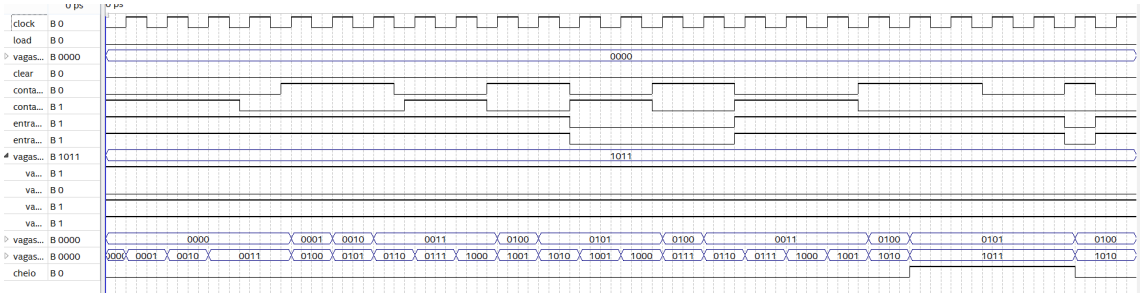
```vhdl
signal vagas_total_s, vagas_idosos_s : std_logic_vector (3 downto 0);
signal entra_sai_total_s, conta_total_s, conta_idosos_s, cheio_s : std_logic;


begin

    entra_sai_total_s <= entra_sai_idosos OR entra_sai_normal;
    conta_total_s <= (conta_idosos OR conta_normal) AND (cheio_s NAND entra_sai_total_s
    conta_idosos_s <= conta_idosos AND (cheio_s NAND entra_sai_idosos);

    cont_idosos: up_down port map (
        clock    => clock,
        zera     => clear,
        conta    => conta_idosos_s,
        carrega  => load,
        entra_sai    => entra_sai_idosos,
        entrada  => vagas_load,
        contagem => vagas_idosos_s
    );

    cont_total: up_down port map (
        clock    => clock,
        zera     => clear,
        conta    => conta_total_s,
        carrega  => load,
        entra_sai    => entra_sai_total_s,
        entrada  => vagas_load,
        contagem => vagas_total_s
    );
```

```
65      comp1: comparador  port map (
66          A     => vagas_total_s,
67          B     => vagas_total,
68          maior_igual => cheio_s
69      );
70
71
72      vagas_out_total <= vagas_total_s;
73      vagas_out_idosos <= vagas_idosos_s;
74      cheio <= cheio_s;
75
76  end arch;
```

iii) Diagrama Lógico



iv) Carta de Tempos



v) Tabela de Testes

clear	conta_idosos	conta_normal	load	Subida	Ciclos	entra_sai_normal	entra_sai_idosos	vagas_out_total	vagas_out_idosos	cheio
1	X	1	X	↑	-	0	0	0000	0000	0
0	1		0	-	0	0	0	0000	0000	0
					1	1	0	0001	0000	0

					2	1	0	0010	0000	0
					3	1	0	0011	0000	0
					4	1	1	0100	0001	0
					5	1	1	0101	0010	0
					6	1	1	0110	0011	0
					7	1	1	0111	0100	0
					8	1	1	1000	0101	1
					9	1	1	1000	0101	1
					10	1	1	1000	0101	1
					11	0	0	0111	0100	0
	0				12	0	0	0110	0011	0
		0			13	1	1	0111	0011	0
	1				14	1	1	1000	0100	1
		1			15	1	1	1000	0100	1

c) Projeto de uma Unidade de Controle em VHDL

i) Descrição funcional

O circuito recebe 7 entradas de 1 bit cada (entra_sai_idosos, entra_sai_normal, conta_idosos, conta_normal, clock, clear, cheio_dp_sm) e retorna 5 saídas de 1 bit cada (entra_sai_idosos_sm_dp, entra_sai_normal_sm_dp, conta_idosos_sm_dp, conta_normal_sm_dp, clear_sm_dp).

ii) Código em VHDL

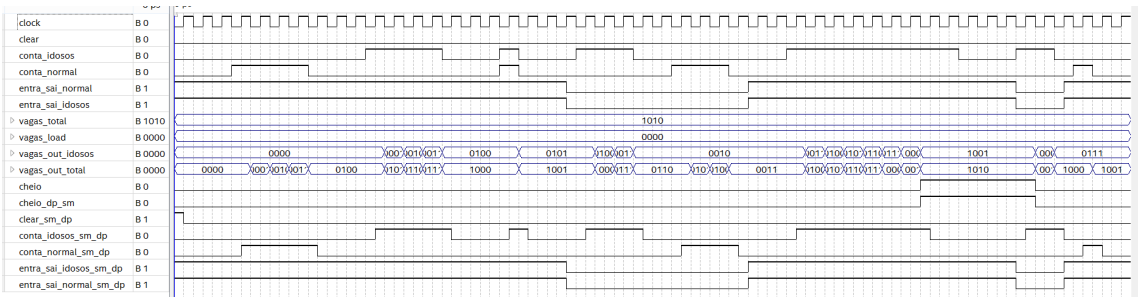
```vhdl
-- library declaration
library IEEE;
use IEEE.std_logic_1164.all;

-- entity
entity unidade_de_controle is
    port(
        entra_sai_idosos, entra_sai_normal, conta_idosos, conta_normal : in  std_logic;
        clock, clear : in  std_logic;
        cheio_dp_sm : in std_logic;
        entra_sai_idosos_sm_dp, entra_sai_normal_sm_dp, conta_idosos_sm_dp, conta_normal_sm_dp, clear_sm_dp: out std_logic
    );
end unidade_de_controle;

-- architecture
architecture arch of unidade_de_controle is
    type state_type is (zera_vagas, vazio, inc_idosos, inc_normal, cheio);
    signal CurrentState, NextState : state_type;
begin
    update_state: process (clock, clear) --talvez dê errado
    begin
        if (clear = '1') then
            CurrentState <= zera_vagas;
        elsif (rising_edge(clock)) then
            CurrentState <= NextState;
        end if;
    end process update_state;
```

```vhdl
    update_output: process (CurrentState)
    begin
        case CurrentState is
            when zera_vagas =>
                clear_sm_dp <= '1';
                conta_idosos_sm_dp <= '0';
                conta_normal_sm_dp <= '0';
                entra_sai_idosos_sm_dp <= 'X'; --talvez dê errado
                entra_sai_normal_sm_dp <= 'X';
            when vazio =>
                clear_sm_dp <= '0';
                conta_idosos_sm_dp <= '0';
                conta_normal_sm_dp <= '0';
                entra_sai_idosos_sm_dp <= 'X'; --talvez dê errado
                entra_sai_normal_sm_dp <= 'X';
            when inc_idosos =>
                clear_sm_dp <= '0';
                conta_idosos_sm_dp <= '1';
                conta_normal_sm_dp <= '0';
                entra_sai_idosos_sm_dp <= entra_sai_idosos; --talvez dê errado
                entra_sai_normal_sm_dp <= entra_sai_idosos;
            when inc_normal =>
                clear_sm_dp <= '0';
                conta_idosos_sm_dp <= '0';
                conta_normal_sm_dp <= '1';
                entra_sai_idosos_sm_dp <= 'X'; --talvez dê errado
                entra_sai_normal_sm_dp <= entra_sai_normal;
```

```
56              when cheio =>
57                  clear_sm_dp <= '0';
58                  conta_idosos_sm_dp <= '0';
59                  conta_normal_sm_dp <= '0';
60                  entra_sai_idosos_sm_dp <= 'X'; --talvez dê errado
61                  entra_sai_normal_sm_dp <= 'X';
62              end case;
63          end process update_output;
64
65          update_next_state: process(entra_sai_idosos, entra_sai_normal, conta_idosos, conta_normal,cheio_dp_sm )
66          begin
67              case CurrentState is
68              when zera_vagas =>
69                  NextState <= vazio;
70              when vazio =>
71                  IF (conta_idosos = '1') then
72                      NextState <= inc_idosos;
73                  ELSIF (conta_normal = '1') then
74                      NextState <= inc_normal;
75                  ELSE
76                      NextState <= vazio;
77                  END IF;
78              when inc_idosos =>
79                  IF (cheio_dp_sm = '1') then
80                      NextState <= cheio;
81                  ELSIF (conta_idosos = '1') then
82                          NextState <= inc_idosos;
83                       ELSE
84                      NextState <= vazio;
85                  END IF;
```

```
86              when inc_normal =>
87                  IF (cheio_dp_sm = '1') then
88                      NextState <= cheio;
89                  ELSIF (conta_normal = '1') then
90                          NextState <= inc_normal;
91                       ELSE
92                      NextState <= vazio;
93                  END IF;
94              when cheio =>
95                  IF (conta_idosos = '1' AND entra_sai_idosos = '0') then --saindo idoso
96                      NextState <= inc_idosos;
97                  ELSIF (conta_normal = '1' AND entra_sai_normal = '0') then --saindo normal
98                      NextState <= inc_normal;
99                  ELSE
100                     NextState <= cheio;
101                 END IF;
102             end case;
103         end process update_next_state;
104     end arch;
```

iii) Carta de Tempos



iv) Tabela de Testes

clear	Subida	Ciclos	conta_idosos	conta_normal	entra_sai_normal	entra_sai_dosos	vagas_total	vagas_out_idosos	vagas_out_total	cheio
1	↑	-	0	0	1	1	1010	0000	0000	0
0	-	0	0	1	1	1	1010	0000	0001	0
		1	0	1	1	1	1010	0000	0010	0
		2	0	1	1	1	1010	0000	0011	0
		3	1	0	1	1	1010	0001	0100	0
		4	1	0	1	1	1010	0010	0101	0
		5	1	0	1	1	1010	0011	0110	0
		6	0	0	1	1	1010	0011	0101	0
		7	1	0	0	0	1010	0010	0100	0
		8	0	0	0	0	1010	0010	0100	0
		9	1	1	1	1	1010	0011	0101	0
		10	1	1	1	1	1010	0100	0110	0
		11	0	1	1	1	1010	0100	0111	0
		12	0	1	1	1	1010	0100	1000	0
		13	1	1	1	1	1010	0101	1001	0
		14	0	1	1	1	1010	0101	1010	1
		15	1	1	1	1	1010	0101	1010	1
		16	1	1	0	0	1010	0100	1001	0

4. Relatório

[Esta seção detalha os resultados obtidos durante a experiência]

a) Resultados obtidos

[Descrever detalhadamente com dados os resultados obtidos baseados no objetivo da experiência. As tabelas de testes planejadas devem estar preenchidas com as saídas obtidas]

Apêndices

[Esta seção apresenta elementos complementares da documentação, como por exemplo, os diagramas lógicos detalhados (esquemáticos no formato padrão), códigos VHDL, cartas de tempo (formas de onda da simulação) ou ainda, informação adicional do projeto desenvolvido]

Referências

1. Apostilas e documentos fornecidos para a experiência.
2. Apostilas disponíveis na plataforma e-Disciplinas.