Disciplina: PCS 3335 – Laboratório Digital A	
Prof.: *Glauber De Bona*	**Data:** *22/03*
Turma: *Glauber - T04*	**Bancada:** *08*
Membros:	
11261531 - Enzo Bustos Da Silva	
10379694 - Davi Augusto Bandeira	

Experiência 06
Máquina de Estados em VHDL

1. Introdução

A experiência 6 do Laboratório Digital visa introduzir o aluno ao conceito de máquina de estados VHDL, com o fito de utilizá-la para sintetizar circuitos no FPGA.

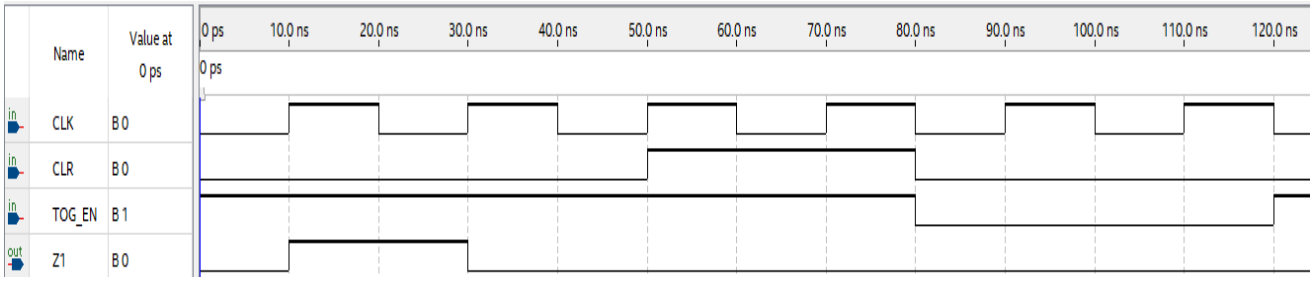
Desta forma, será implementado uma nova entrada e saída do circuito da experiência anterior para simular a possibilidade da entrada/saída de um idoso, o qual deve ser tomado como prioridade para acesso ao estacionamento.

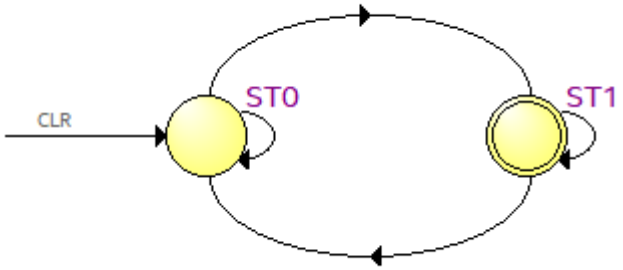
2. Objetivo

O objetivo a ser alcançado no final da experiência consiste no aprendizado da implementação de máquina de estados em VHDL.

3. Planejamento

3.1 Projeto de uma Máquina de Estados em VHDL





Observando o diagrama gerado pelo State Machine Viewer, em comparação com o diagrama de estados fornecido pelos professores, percebemos que não existem legendas que indicam satisfatoriamente quando ocorre uma mudança ou permanência de estados. Além disso, não se sabe qual é o valor da saída em cada dos estados apenas olhando para este diagrama. Contudo, fora essas diferenças estéticas, o diagrama é o mesmo.

```
def maquina_de_estados(tog_en, clk, clr):
    Z = "0" #Output pré-definido
    present_state = "0"

    #Se o clear estiver ligado, vamos direto ao Estado 0
    if clr == "1":
        present_state = "0"

    #Se tog_en estiver em 0 e estivermos no Estado 0, continuamos no Estado 0
    elif rising_edge(clk) and tog_en == "0" and present_state == "0":
        present_state = "0"

    #Se tog_en estiver em 1 e estivermos no Estado 0, vamos para o Estado 1
    elif rising_edge(clk) and tog_en == "1" and present_state == "0":
        present_state = "1"

    #Se tog_en estiver em 0 e estivermos no Estado 1, continuamos no Estado 1
    elif rising_edge(clk) and tog_en == "0" and present_state == "1":
        present_state = "1"

    #Se tog_en estiver em 1 e estivermos no Estado 0, vamos para o Estado 0
    elif rising_edge(clk) and tog_en == "1" and present_state == "1":
        present_state = "0"

    #Captura qualquer outro caso
    else:
        present_state = "0"

    #Atribuição da Saída, dependendo do Estado que estamos
    if present_state == "0":
        Z = "0"

    elif present_state == "1":
        Z = "1"

    return Z
```

3.2 Projeto de Fluxo de dados em VHDL

a) Código

O Código em VHDL para o projeto com o Fluxo de dados do estacionamento que também comporta vagas para idosos segue abaixo:

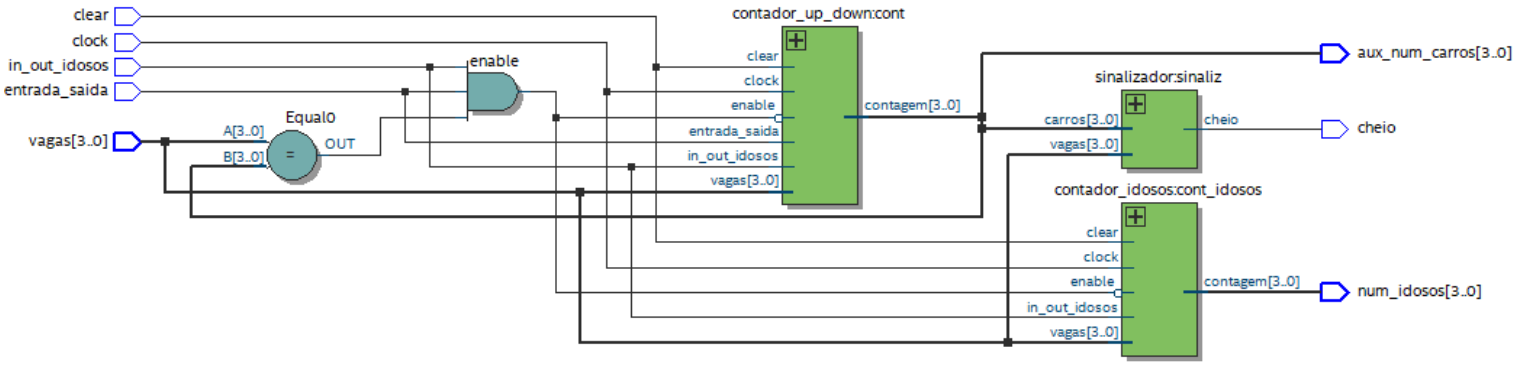
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity projeto is
      port(
         clock, clear, entrada_saida, in_out_idosos: in std_logic;
         vagas: in std_logic_vector (3 downto 0);
         num_idosos: out std_logic_vector (3 downto 0);
         cheio: out std_logic
      );
end projeto;

architecture behavior of projeto is
      signal enable: std_logic;
      signal contagem: std_logic_vector (3 downto 0); -- numero de carros r
      signal lotacao: std_logic; -- alto em caso de lotacao do estacionamer
      signal contagem_idosos: std_logic_vector (3 downto 0); -- numero de i
      signal contagem_int, vagas_int: unsigned (3 downto 0); -- guarda valc

      component contador_up_down is
      port(
         clock, clear, entrada_saida, enable, in_out_idosos: in std_logic;
         vagas: in std_logic_vector (3 downto 0);
         contagem: out std_logic_vector (3 downto 0);
         fim: out std_logic
      );
      end component;

      component sinalizador is
      port(
         vagas, carros: in std_logic_vector (3 downto 0);
         cheio: out std_logic
      );
      end component;

      component contador_idosos is
      port(
         clock, clear, in_out_idosos, enable: in std_logic;
         vagas: in std_logic_vector (3 downto 0);
         contagem: out std_logic_vector (3 downto 0);
         fim: out std_logic
      );
      end component;
begin

   enable <= '0' when (contagem = vagas and entrada_saida ='1' and in_out_idosos = '1') else '1';
   num_idosos <= contagem_idosos;
   contagem_int <= unsigned(contagem);
   vagas_int <= unsigned(vagas);
   num_idosos <= contagem_idosos;

   cont: contador_up_down port map(clock, clear, entrada_saida, enable, in_out_idosos, vagas, contagem, lotacao);
   cont_idosos: contador_idosos port map(clock, clear, in_out_idosos, enable, vagas, contagem_idosos, lotacao);
   sinaliz: sinalizador port map(vagas, contagem, cheio);

end behavior;
```

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

entity contador_idosos is
	port(
		clock, clear, in_out_idosos, enable: in std_logic;
		vagas: in std_logic_vector (3 downto 0);
		contagem: out std_logic_vector (3 downto 0);
		fim: out std_logic
	);
end contador_idosos;

architecture arc_cont of contador_idosos is
	signal IQ: integer range 0 to 15;
	signal limite: integer range 0 to 15;
begin
	limite <= to_integer(unsigned(vagas)); -- converte vagas para um numero inteiro a ser comparado no if process

	process(clock, clear, in_out_idosos, enable, IQ)
	begin
		if clear = '1' then IQ <= 0;
		elsif rising_edge(clock) then
			if in_out_idosos = '1' and (IQ < limite) and enable = '1' then -- existem vagas disponiveis e um carro entra, cas
				IQ <= IQ + 1;
			elsif in_out_idosos = '0' and (IQ > 0) and enable = '1' then -- algum carro sai, dado que existe algum carro
				IQ <= IQ - 1;
			end if;
		end if;
	end process;

	contagem <= std_logic_vector(to_unsigned(IQ, contagem'length));


	fim <= '1' when IQ = 15 else '0';

end arc_cont;
```

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

entity contador_up_down is
	port(
		clock, clear, entrada_saida, enable, in_out_idosos: in std_logic;
		vagas: in std_logic_vector (3 downto 0);
		contagem: out std_logic_vector (3 downto 0);
		fim: out std_logic
	);
end contador_up_down;

architecture behavior of contador_up_down is
	signal IQ: integer range 0 to 15;
	signal limite: integer range 0 to 15;
begin
	limite <= to_integer(unsigned(vagas)); -- converte vagas para um numero inteiro a ser comparado no i

	process(clock, clear, entrada_saida, enable, IQ)
	begin
		if clear = '1' then IQ <= 0;
		elsif rising_edge(clock) then
		-- entrada de carro normal, saida de idoso (nada acontece)
			if entrada_saida = '1' and enable = '1' and in_out_idosos = '0' then
				IQ <= IQ;
		-- saida de ambos tipos de carro
			elsif entrada_saida = '0' and (IQ > 1) and enable = '1' and in_out_idosos = '0' then
				IQ <= IQ - 2;
		-- saida de ambos tipos de carro caso so tenha 1 dentro
			elsif entrada_saida = '0' and (IQ = 1) and enable = '1' and in_out_idosos = '0' then
				IQ <= IQ - 1;
		-- entrada de carro normal, entrada de idoso caso haja 2 ou mais vagas
			elsif entrada_saida = '1' and (IQ < (limite-1)) and enable = '1' and in_out_idosos = '1' then
				IQ <= IQ + 2;
		-- entrada de ambos tipos de carro, caso so haja 1 vaga, so permitir a entrada de idoso
			elsif entrada_saida = '1' and (IQ = (limite-1)) and enable = '1' and in_out_idosos = '1' then
				IQ <= IQ + 1;
		-- saida de carro normal, entrada de idoso (nada acontece)
			elsif entrada_saida = '0' and enable = '1' and in_out_idosos = '1' then
				IQ <= IQ;

			end if;
		end if;
	end process;

	contagem <= std_logic_vector(to_unsigned(IQ, contagem'length));


	fim <= '1' when IQ = 15 else '0';

end behavior;
```

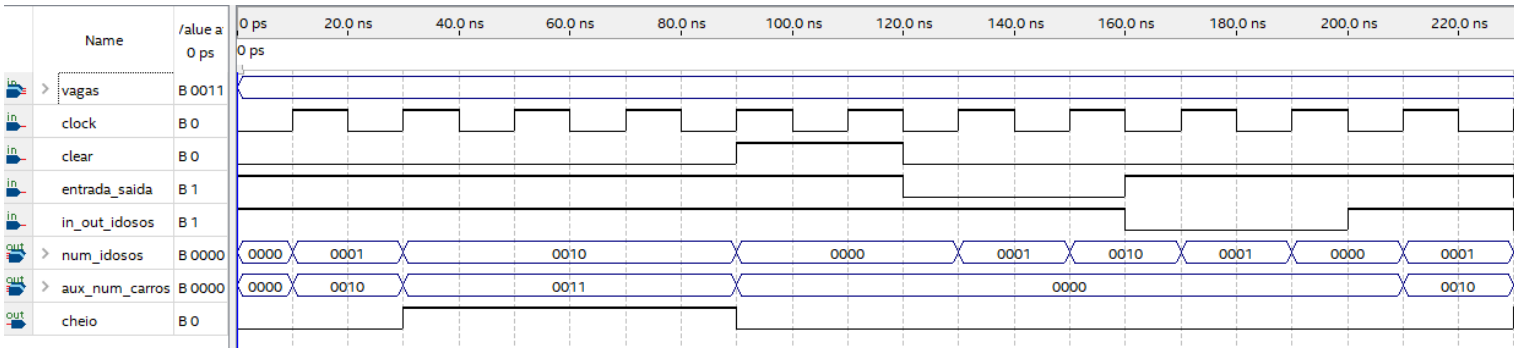
b) RTL viewer do circuito

Através da plataforma RTL Viewer do Quartus, pudemos gerar a seguinte imagem das interconexões que ocorrem no nosso circuito lógico, podemos confirmar que as conexões estão corretamente feitas.



c) Carta de Tempos

Através da simulação do circuito gerando um arquivo de Waveform, conseguimos obter a carta de tempos mostrada a seguir, essa carta de tempos foi convertida na tabela de testes do nosso circuito que será utilizada no dia do experimento.



d) Tabela de Testes

Segue a tabela de testes que foi gerada a partir da carta de tempos do circuito, essa tabela será utilizada para validação da implementação feita quando passarmos o circuito para a placa de FPGA:

Número de vagas = "0011" (3 Vagas)					
Clock	Clear	Entrada_Saída	In_Out_Idosos	Número de Idosos	Cheio
1↑	0	1	1	0	0
2↑	0	1	1	1	1

3↑	0	1	1	2	1
4↑	0	1	1	2	1
5↑	1	1	1	0	0
6↑	1	1	1	0	0
7↑	0	0	1	1	0
8↑	0	0	1	2	0
9↑	0	1	0	1	0
10↑	0	1	0	0	0
11↑	0	1	1	1	0

3.3 Projeto de uma Unidade de Controle em VHDL

O projeto da Unidade de Controle (UC) não foi necessário de ser implementado, pois a UC já estava implementada no Fluxo de Dados (FD), isso foi conversado e concordado com o professor, pois todo o circuito da FD já implementou todo o circuito necessário

4. Relatório

A implementação do circuito durante o experimento 6 em laboratório teve que ser adaptada minimamente para adequação à placa FPGA, a mudança foi basicamente que aumentamos o número de chave, tendo 4 chaves: 1 para entrada de carros normais, 1 para saída de carros normais, 1 para entrada de carros de idosos e 1 para saída de carros de idoso; essa mudança foi comentada e explicada com o professor Glauber que aprovou a mudança para uma melhor execução do projeto. Além da placa FPGA, foi utilizado o software WaveForms e o dispositivo Analog Discovery.

A designação de pinos na placa FPGA, pode ser conferida na tabela abaixo:

Node Name	Location
cheio	PIN_AA2
clear	PIN_U13
clock	PIN_A12
vagas[3]	PIN_AB12
vagas[2]	PIN_AB13
vagas[1]	PIN_AA13
vagas[0]	PIN_AA14
num_idosos[0]	PIN_U2
num_idosos[1]	PIN_U1
num_idosos[2]	PIN_L2
num_idosos[3]	PIN_L1
aux_contagem[3]	PIN_N2
aux_contagem[2]	PIN_Y3
aux_contagem[1]	PIN_W2
aux_contagem[0]	PIN_AA1
entrada	PIN_V13
saida	PIN_T13
in_idosos	PIN_T12
out_idosos	PIN_AA15

Seguem os códigos adaptados dos módulos utilizados no projeto em laboratório.

- sinalizador.vhd:

```
library IEEE;

use IEEE.std_logic_1164.all;



entity sinalizador is

        port(

            vagas, carros: in std_logic_vector (3 downto 0);

            cheio: out std_logic

        );

end sinalizador;



architecture behavior of sinalizador  is

begin



    cheio <= '1' when vagas = carros else '0';



end behavior;
```

- contador_up_down.vhd:

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

entity contador_up_down is
	port(
		clock, clear, entrada, saida, enable, in_idosos, out_idosos: in std_logic;
		vagas: in std_logic_vector (3 downto 0);
		contagem, contagem_idosos, contagem_normal: out std_logic_vector (3 downto 0);
		fim: out std_logic
	);
end contador_up_down;

architecture behavior of contador_up_down is
	signal IQ_IDOSO, IQ_NORMAL, IQ: integer range 0 to 15;
	signal limite: integer range 0 to 15;
begin
	limite <= to_integer(unsigned(vagas)); -- converte vagas para um numero inteiro a ser
comparado no if process

	process(clock, clear, entrada, saida, in_idosos, out_idosos, enable, IQ)
	begin
		if clear = '1' then
			IQ <= 0;
			IQ_IDOSO <= 0;
			IQ_NORMAL <= 0;
```

```
            elsif rising_edge(clock) and enable = '1' then


                if entrada = '0' and saida = '0' and in_idosos = '0' and out_idosos = '0'
then

                  IQ <= IQ;

                  IQ_IDOSO <= IQ_IDOSO;

                  IQ_NORMAL <= IQ_NORMAL;


                elsif entrada = '0' and saida = '0' and in_idosos = '0' and out_idosos = '1'
and (IQ_IDOSO > 0) then

                  IQ <= IQ - 1;

                  IQ_IDOSO <= IQ_IDOSO - 1;

                  IQ_NORMAL <= IQ_NORMAL;


                elsif entrada = '0' and saida = '0' and in_idosos = '1' and out_idosos = '0'
and (IQ < limite) then

                  IQ <= IQ + 1;

                  IQ_IDOSO <= IQ_IDOSO + 1;

                  IQ_NORMAL <= IQ_NORMAL;


                elsif entrada = '0' and saida = '0' and in_idosos = '1' and out_idosos = '1'
then

                  IQ <= IQ;

                  IQ_IDOSO <= IQ_IDOSO;

                  IQ_NORMAL <= IQ_NORMAL;


                elsif entrada = '0' and saida = '1' and in_idosos = '0' and out_idosos = '0'
and (IQ_NORMAL > 0) then

                  IQ <= IQ - 1;
```

```
                    IQ_IDOSO <= IQ_IDOSO;

                    IQ_NORMAL <= IQ_NORMAL - 1;


                elsif entrada = '0' and saida = '1' and in_idosos = '0' and out_idosos = '1'
and (IQ_NORMAL > 0) and (IQ_IDOSO > 0) then

                    IQ <= IQ - 2;

                    IQ_IDOSO <= IQ_IDOSO - 1;

                    IQ_NORMAL <= IQ_NORMAL - 1;


                elsif entrada = '0' and saida = '1' and in_idosos = '1' and out_idosos = '0'
and (IQ_NORMAL > 0) then

                    IQ <= IQ;

                    IQ_IDOSO <= IQ_IDOSO + 1;

                    IQ_NORMAL <= IQ_NORMAL - 1;


                elsif entrada = '0' and saida = '1' and in_idosos = '1' and out_idosos = '1'
and (IQ_NORMAL > 0) then

                    IQ <= IQ - 1;

                    IQ_IDOSO <= IQ_IDOSO;

                    IQ_NORMAL <= IQ_NORMAL - 1;


                elsif entrada = '1' and saida = '0' and in_idosos = '0' and out_idosos = '0'
and (IQ < limite) then

                    IQ <= IQ + 1;

                    IQ_IDOSO <= IQ_IDOSO;

                    IQ_NORMAL <= IQ_NORMAL + 1;


                elsif entrada = '1' and saida = '0' and in_idosos = '0' and out_idosos = '1'
and (IQ_IDOSO > 0)then
```

```
                    IQ <= IQ;

                    IQ_IDOSO <= IQ_IDOSO - 1;

                    IQ_NORMAL <= IQ_NORMAL + 1;


                elsif entrada = '1' and saida = '0' and in_idosos = '1' and out_idosos = '0'
and (IQ < limite - 1) then

                    IQ <= IQ + 2;

                    IQ_IDOSO <= IQ_IDOSO + 1;

                    IQ_NORMAL <= IQ_NORMAL + 1;


                elsif entrada = '1' and saida = '0' and in_idosos = '1' and out_idosos = '0'
and (IQ = limite - 1) then

                    IQ <= IQ + 1;

                    IQ_IDOSO <= IQ_IDOSO + 1;

                    IQ_NORMAL <= IQ_NORMAL;


                elsif entrada = '1' and saida = '0' and in_idosos = '1' and out_idosos = '1'
and (IQ < limite) then

                    IQ <= IQ + 1;

                    IQ_IDOSO <= IQ_IDOSO;

                    IQ_NORMAL <= IQ_NORMAL + 1;


                elsif entrada = '1' and saida = '1' and in_idosos = '0' and out_idosos = '0'
then

                    IQ <= IQ;

                    IQ_IDOSO <= IQ_IDOSO;

                    IQ_NORMAL <= IQ_NORMAL;
```

```
                elsif entrada = '1' and saida = '1' and in_idosos = '0' and out_idosos = '1'
and (IQ_IDOSO > 0) then

                    IQ <= IQ - 1;

                    IQ_IDOSO <= IQ_IDOSO - 1;

                    IQ_NORMAL <= IQ_NORMAL;


                elsif entrada = '1' and saida = '1' and in_idosos = '1' and out_idosos = '1'
then

                    IQ <= IQ;

                    IQ_IDOSO <= IQ_IDOSO;

                    IQ_NORMAL <= IQ_NORMAL;


                end if;

            end if;

        end process;


        contagem <= std_logic_vector(to_unsigned(IQ, contagem'length));

        contagem_idosos <= std_logic_vector(to_unsigned(IQ_IDOSO, contagem'length));

        contagem_normal <= std_logic_vector(to_unsigned(IQ_NORMAL, contagem'length));


        fim <= '1' when IQ = 15 else '0';


end behavior;
```

- projeto.vhd:

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

entity projeto is
        port(
            clock, clear, entrada, saida, in_idosos, out_idosos: in std_logic;
            vagas: in std_logic_vector (3 downto 0);
            num_idosos: out std_logic_vector (3 downto 0);
            cheio: out std_logic;
            saida_hex: out std_logic_vector (6 downto 0);
            aux_contagem: out std_logic_vector (3 downto 0)
        );
end projeto;

architecture behavior of projeto is
        signal enable: std_logic;
        signal sig_saida_hex: std_logic_vector (6 downto 0);
         signal contagem, contagem_idosos, contagem_normal: std_logic_vector (3 downto 0); --
numero de carros no contador
        signal lotacao: std_logic; -- alto em caso de lotacao do estacionamento

        component contador_up_down is
        port(
            clock, clear, entrada, saida, enable, in_idosos, out_idosos: in std_logic;
            vagas: in std_logic_vector (3 downto 0);
```

```
            contagem, contagem_idosos, contagem_normal: out std_logic_vector (3 downto 0);
            fim: out std_logic
        );
        end component;

        component sinalizador is
        port(
            vagas, carros: in std_logic_vector (3 downto 0);
            cheio: out std_logic
        );
        end component;

        component contador_idosos is
        port(
            clock, clear, in_idosos, out_idosos, enable: in std_logic;
            vagas: in std_logic_vector (3 downto 0);
            contagem: out std_logic_vector (3 downto 0);
            fim: out std_logic
        );
        end component;
begin

    enable <= '0' when (contagem = vagas and ((entrada = '1' and saida ='0') or (in_idosos =
'1' and out_idosos = '0'))) else '1';
    num_idosos <= contagem_idosos;

       cont:  contador_up_down  port  map(clock,  clear,  entrada,  saida,  enable,  in_idosos,
out_idosos, vagas, contagem,
```

```
    contagem_idosos, contagem_normal, lotacao);

    sinaliz: sinalizador port map(vagas, contagem, cheio);



    aux_contagem <= contagem;

    process (sig_saida_hex, contagem_normal) begin

        if (contagem_normal = "0000") then

            sig_saida_hex <= "1000000";

        elsif (contagem_normal = "0001") then

            sig_saida_hex <= "1111001";

        elsif (contagem_normal = "0010") then

            sig_saida_hex <= "0100100";

        elsif (contagem_normal = "0011") then

            sig_saida_hex <= "0110000";

        elsif (contagem_normal = "0100") then

            sig_saida_hex <= "0011001";

        elsif (contagem_normal = "0101") then

            sig_saida_hex <= "0010010";

        elsif (contagem_normal = "0110") then

            sig_saida_hex <= "0000010";

        elsif (contagem_normal = "0111") then

            sig_saida_hex <= "1111000";

        elsif (contagem_normal = "1000") then

            sig_saida_hex <= "0000000";

        elsif (contagem_normal = "1001") then

            sig_saida_hex <= "0010000";

        end if;

    end process;
```

```
    saida_hex <= sig_saida_hex;


end behavior;
```

5. Desafio

Para o desafio era necessário utilizar o display de 7 segmentos para mostrar o número de carros normais que estavam utilizando o estacionamento, no código acima já está com essa implementação, mas basicamente já tínhamos o controle desse sinal com a variável contagem_normal, então basicamente precisamos adicionar um pedaço de código que convertia a contagem para o display de 7 segmentos, fizemos isso com o seguinte código:

```
    process (sig_saida_hex, contagem_normal) begin

        if (contagem_normal = "0000") then

            sig_saida_hex <= "1000000";

        elsif (contagem_normal = "0001") then

            sig_saida_hex <= "1111001";

        elsif (contagem_normal = "0010") then

            sig_saida_hex <= "0100100";

        elsif (contagem_normal = "0011") then

            sig_saida_hex <= "0110000";

        elsif (contagem_normal = "0100") then

            sig_saida_hex <= "0011001";

        elsif (contagem_normal = "0101") then

            sig_saida_hex <= "0010010";

        elsif (contagem_normal = "0110") then

            sig_saida_hex <= "0000010";

        elsif (contagem_normal = "0111") then

            sig_saida_hex <= "1111000";
```

```
        elsif (contagem_normal = "1000") then

            sig_saida_hex <= "0000000";

        elsif (contagem_normal = "1001") then

            sig_saida_hex <= "0010000";

        end if;

    end process;


    saida_hex <= sig_saida_hex;
```