

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 05

Introdução ao LabDig A

1. Introdução

Na experiência 6 do Laboratório Digital visa introduzir o aluno à linguagem VHDL, com o fito de utilizá-la para sintetizar circuitos no FPGA. Desta forma, será realizado um projeto de circuito onde os alunos irão, inicialmente, simulá-lo em VHDL no planejamento e, em seguida, implantá-lo no FPGA para o relatório da experiência.

2. Objetivo

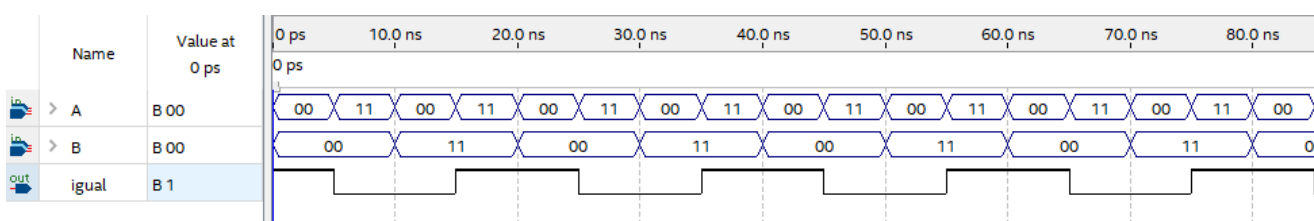
O objetivo deste experimento é montar um projeto de estacionamento, no qual o operador impõe o número de vagas disponíveis (entrada “vagas”), a entrada e/ou saídas de carros (incrementando ou decrementando um contador) e a possibilidade de limpar todas as vagas (comando “clear”). Ademais, teremos uma saída “cheio” que estará em alto caso o número de carros presente seja igual ao número de vagas definido.

Para formar este projeto, é utilizada uma top entity que engloba duas entidades menores, a de um contador up/down de 4 bits e um sinalizador.

3. Planejamento

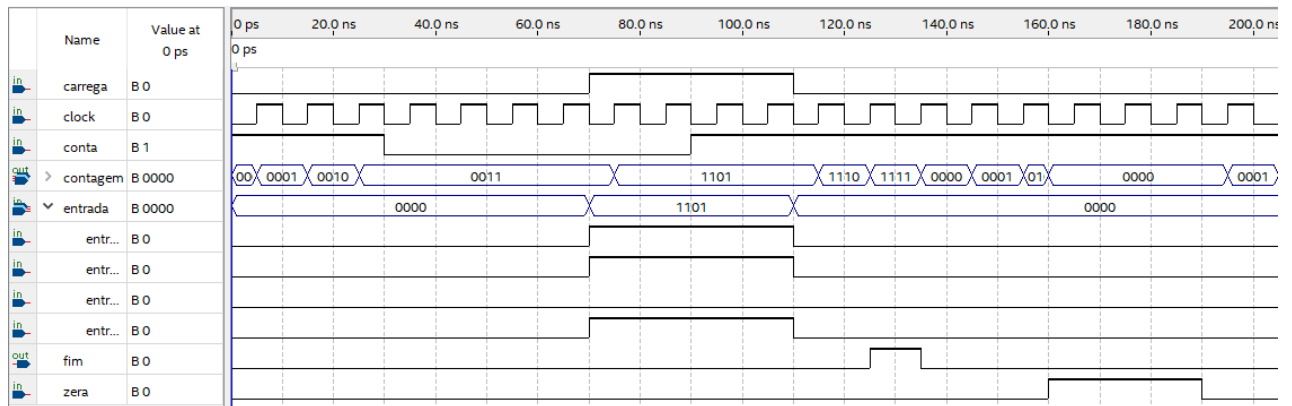
3.2 Projeto de um Comparador e um Contador em VHDL

a) Comparador binário



Nessa simulação do comparador binário, foi colocado duas entradas de forma que o clock de “A” é duas vezes mais rápido que o de “B” para que fosse observado a dinâmica da arquitetura funcionando. Desta forma, é evidente que a saída “igual” opera corretamente, visto que é igual a 1 nos momentos em que $A=B$, como esperado.

b) Contador hexadecimal



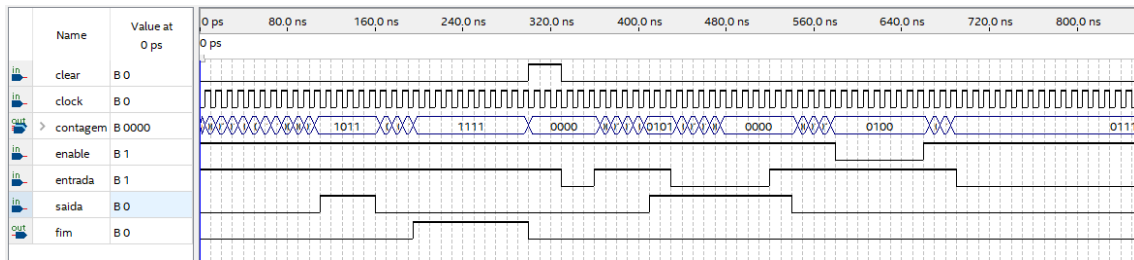
Nesta simulação do contador é possível observar o comportamento de todas as entradas (isolada e conjuntamente) para testar se a arquitetura do código em VHDL está operando de acordo com o esperado.

De fato, quando temos a entrada “conta” em alto, teremos um incremento na saída “contagem” a cada borda de subida do clock. No entanto, é importante ressaltar que quando se tem a entrada “carrega” em alto, a saída “contagem” receberá o vetor “entrada”, nas subidas do clock, independentemente de “conta”, isto é, não ocorre incremento e a saída se torna fixada, como visto nos instantes entre 90 a 110 ns. Ademais, situação semelhante acontece com o comando “zera”, afinal este impõe a saída “contagem” para 0 e, quando está ativo, a entrada “conta” não tem efeito.

Finalmente, podemos observar também que ao ser alcançada a contagem limite, a saída “fim” fica em alto durante esse período.

3.3 Projeto Estrutural em VHDL

a) Contador up/down de 4 bits



Nesta simulação do contador up/down de 4 bits é possível observar seu comportamento. São adicionadas 2 entradas ao circuito “entrada” e “saída”, indicando a entrada e/ou saída de carros, isto é, quando cada uma está em alto, implica na incremento e/ou decremento. É evidente que, se ambas estão em baixo ou alto, o contador não altera.

Uma entrada “enable” também foi adicionada a este contador, de forma que, caso esteja em baixo, o contador permanece fixado. O motivo para esse sinal de entrada ao contador é porque o “enable” será usado como sinal intermediário para o projeto na top entity, de tal forma que ele será uma realimentação vinda do sinalizador para indicar que o estacionamento se encontra cheio, consequentemente impedindo a nova entrada de carros, isto é, parando o contador.

b) Código da top-level entity englobando o contador up/down e o sinalizador

```

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

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

architecture behavior of projeto is
    signal enable: std_logic; -- feedback do sinalizador para o contador saber quando i
    signal contagem: 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 std_logic;
            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;

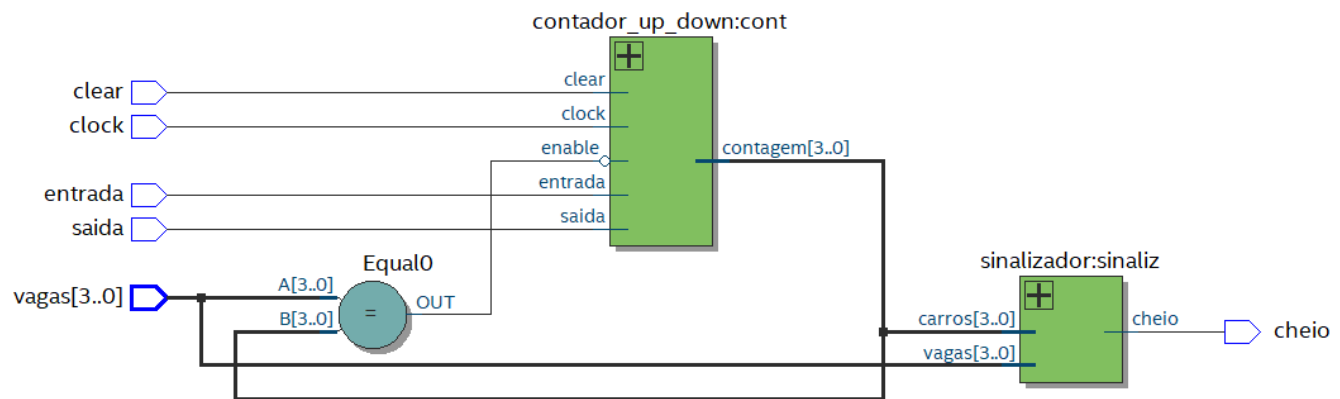
begin
    enable <= '0' when contagem = vagas else '1';

    cont: contador_up_down port map(clock, clear, entrada, saida, enable, contagem, lot
    sinaliz: sinalizador port map(vagas, contagem, cheio);

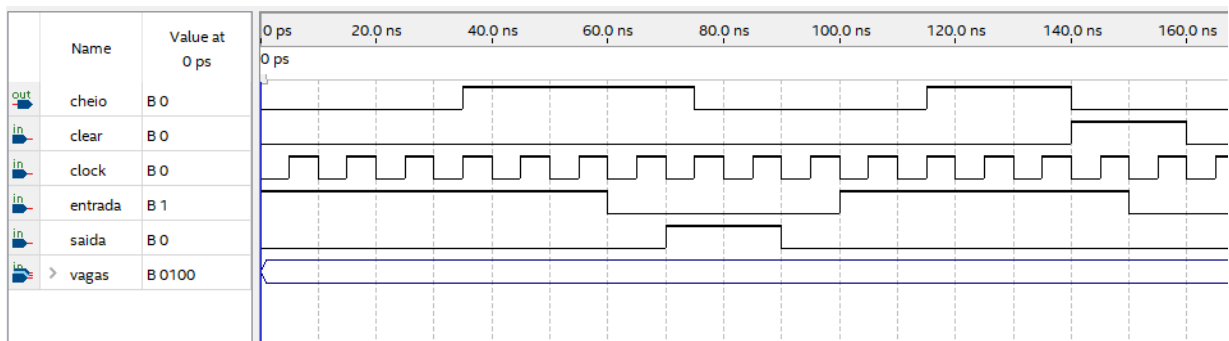
end behavior;

```

c) RTL Viewer do projeto



d) Simulação do projeto



Nesta simulação do projeto é possível observar seu comportamento. Foram definidas 4 vagas ao estacionamento na entrada “vagas”.

Após 4 bordas de subida do clock enquanto “entrada” = 1, de fato observamos a saída “cheio” em alto, pois a entrada “vagas” = 0100. É interessante observar que, após o estacionamento estar cheio, “entrada” continuou em alto por mais algumas bordas de clock, mas isso não implicou em uma alteração na contagem interna. Afinal, ao fazermos “saída” = 1, a saída “cheio” imediatamente pra baixo assim que ocorre o evento do clock, indicando que o estacionamento, de fato, não aceita carros quando se encontra lotado, pois sua contagem estabiliza enquanto a saída “cheio” = 1.











A tabela de testes do circuito, já preenchida (após o experimento) segue abaixo (o circuito foi testado no dia para 1, 3, 4, 5 e 15 vagas juntamente com o professor):

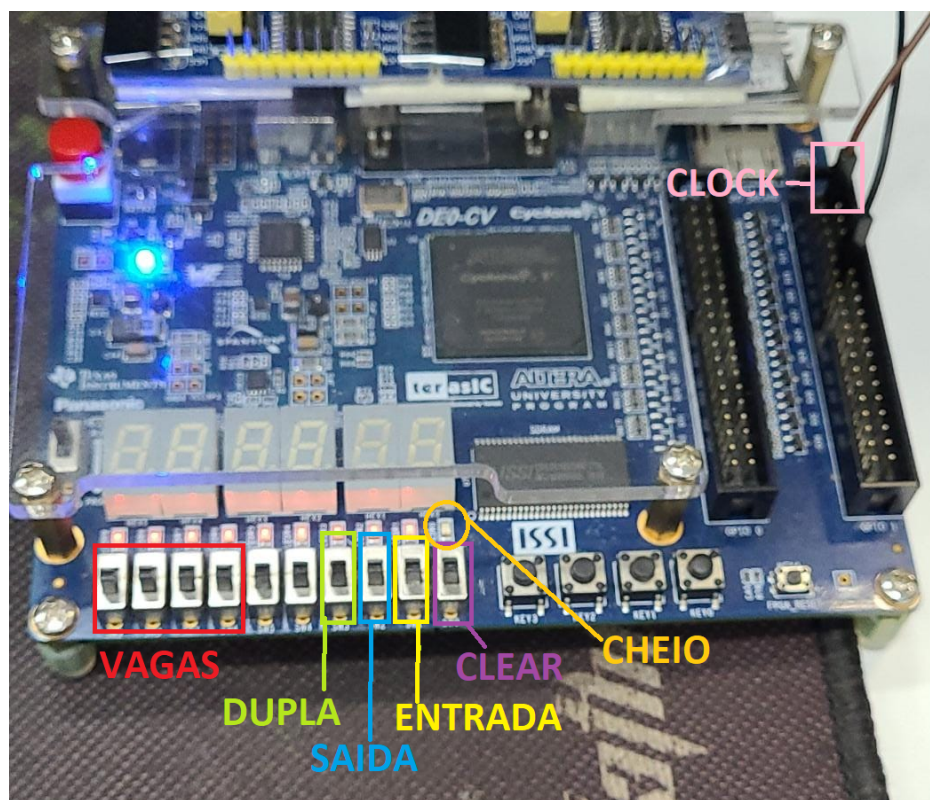
Vagas = "0100" (4 vagas)						
Clock (borda de subida)	Clear	Entrada	Saída	(Número de Carros)	Cheio (Esperado)	Cheio (Obtido)
1	0	1	0	1	0	0
2	0	1	0	2	0	0
3	0	1	0	3	0	0
4	0	1	0	4	1	1
5	0	1	0	4	1	1
6	0	1	0	4	1	1
7	0	0	0	4	1	1
8	0	0	1	3	0	0
9	0	0	1	2	0	0
10	0	0	0	2	0	0
11	0	1	0	3	0	0
12	0	1	0	4	1	1
13	0	1	0	4	1	1
14	0	1	0	4	1	1
15	1	1	0	0	0	0
16	1	0	0	0	0	0
17	0	0	0	0	0	0

3.4 Implementação do Circuito de Vagas de Estacionamento

Durante a experiência no laboratório, o circuito programado foi integrado à placa FPGA e o resultado foi obtido conforme esperado. Ademais, foi utilizado o software WaveForms junto com o dispositivo Analog Discovery para simular o clock do circuito.

Segue abaixo a tabela de designação dos pinos do FPGA:

Node Name	Direction	Location
 bool_dupla	Input	PIN_T12
 cheio	Output	PIN_AA2
 clear	Input	PIN_U13
 clock	Input	PIN_A12
 entrada	Input	PIN_V13
 saida	Input	PIN_T13
 vagas[3]	Input	PIN_AB12
 vagas[2]	Input	PIN_AB13
 vagas[1]	Input	PIN_AA13
 vagas[0]	Input	PIN_AA14



3.4 Desafio

O desafio da experiência consistia em adicionar uma nova entrada para simular a entrada e/ou saída de veículos grandes que ocupam duas vagas do estacionamento.

Desta forma, foi adicionado uma nova entrada booleana na entidade contadora “bool_dupla”, onde sua ativação indicaria a entrada e/ou saída de um veículo que ocupasse 2 vagas e, caso em 0, manteria o caso padrão. Segue abaixo a alteração no código:

```
entity contador_up_down is
  port(
    clock, clear, entrada, saida, enable, bool_dupla: 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 if process
  process(clock, clear, entrada, saida, enable, IQ)
  begin
    if clear = '1' then IQ <= 0;
    elsif clock'event and clock = '1' then
      if entrada = '1' and saida = '0' and (IQ < limite) and enable = '1' and bool_dupla = '0' then -- existem vagas d
        IQ <= IQ + 1;
      elsif entrada = '1' and saida = '0' and (IQ < (limite-1)) and enable = '1' and bool_dupla = '1' then
        IQ <= IQ + 2;
      elsif entrada = '0' and saida = '1' and (IQ > 0) and bool_dupla = '0' then -- algum carro sai, dado que existe a
        IQ <= IQ - 1;
      elsif entrada = '0' and saida = '1' and (IQ > 1) and bool_dupla = '1' then
        IQ <= IQ - 2;
      -- caso entrada e saida ambos igual a 0 ou 1, o contador nao altera
    end if;
  end if;
end process;

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

  fim <= '1' when IQ = 15 else '0';
end behavior;
```

Para o desafio, foram realizados testes em sala para checar se o código operava corretamente em casos extremos, isto é, se negava a entrada de veículos de tamanho 2 quando só se tinha 1 vaga disponível e se negava a saída de veículos de tamanho 2 quando só se havia 1 veículo presente no estacionamento. O resultado saiu como esperado.