Disciplina: PCS 3335 – Laboratório Digital A	
Prof.: *Glauber de Bona*	**Data:** *25/04/2020*
Turma: *4*	**Bancada:** *B8*
Membros:	
9912532 - Rafael Lucchesi Piacente	
10772925 - Gustavo Donnini Chen	

Experiência *6*

Maquinas de Estado em VHDL

1. Introdução

Esta experiência apresenta máquinas de estados descritas em VHDL e sua aplicação em um circuito digital simples. Ao final da experiência, os alunos terão conhecimento sobre o desenvolvimento de sistemas digitais mais complexos, compostos por fluxo de dados e unidade de controle.

2. Objetivo

Após a conclusão desta experiência, os seguintes tópicos devem ser conhecidos pelos alunos:

- Descrição de máquinas de estados em VHDL;
- Aplicação de máquinas de estados como unidade de controle de um circuito digital;
- Estudo de um circuito digital simples.

3. Parte experimental:

1. Projeto de uma Máquina de Estados em VHDL

a. Consideramos a descrição comportamental de uma máquina de estados em VHDL (figura 1).

```
1  -- maquina_estados.vhd
2  --
3
4   library IEEE;
5   use IEEE.std_logic_1164.all;
6
7  entity maquina_estados is
8    port ( clock, reset :          in  std_logic;
9           iniciar, condicao1, fim: in  std_logic;
10          pronto:                 out std_logic;
11          zera1, conta1:          out std_logic;
12          conta2, carrega2:       out std_logic;
13          db_estado:              out std_logic_vector(2 downto 0)
14        );
15 end maquina_estados;
16
17 architecture comportamental of maquina_estados is
18    type Tipo_estado is (inicial, zeraconts, testacondicao1,
19                         contacont2, contacont1, testafim, final);
20    signal Eatual, Eprox: Tipo_estado;
21
22 begin
23
24    -- proximo estado (reset, borda do clock)
25    process (reset, clock)
26    begin
27       if reset = '1' then
28          Eatual <= inicial;
29       elsif clock'event and clock = '1' then
30          Eatual <= Eprox;
31       end if;
32    end process;
33
34    -- proximo estado
35    process (iniciar, condicao1, fim, Eatual)
36    begin
37       case Eatual is
38          when inicial =>          if iniciar='1'
39                                   then Eprox <= zeraconts;
40                                   else Eprox <= inicial;
41                                   end if;
42
43          when zeraconts =>        Eprox <= testacondicao1;
44
45          when testacondicao1 =>   if condicao1='1'
46                                   then Eprox <= contacont2;
47                                   else Eprox <= contacont1;
48                                   end if;
49
50          when contacont2 =>       Eprox <= contacont1;
51
52          when contacont1 =>       Eprox <= testafim;
53
54          when testafim =>         if fim='1'
55                                   then Eprox <= final;
56                                   else Eprox <= testacondicao1;
57                                   end if;
58
59          when final =>            Eprox <= inicial;
60
61          when others =>           Eprox <= inicial;
62       end case;
63    end process;
64
65    -- saidas
66    with Eatual select
67       zera1 <= '1' when zeraconts, '0' when others;
68
69    with Eatual select
70       carrega2 <= '1' when zeraconts, '0' when others;
71
72    with Eatual select
73       conta2 <= '1' when contacont2, '0' when others;
74
75    with Eatual select
76       conta1 <= '1' when contacont1, '0' when others;
77
78    with Eatual select
79       pronto <= '1' when final, '0' when others;
80
81    with Eatual select
82       db_estado <= "000" when inicial,
83                    "001" when zeraconts,
84                    "010" when testacondicao1,
85                    "011" when contacont2,
86                    "100" when contacont1,
87                    "101" when testafim,
88                    "110" when final,
89                    "111" when others;
90
91 end comportamental;
```

Figura 1: descrição comportamental de uma máquina de estados em VHDL.

b. Tendo em vista a descrição dada, desenvolvemos o seguinte diagrama de transição de estados:

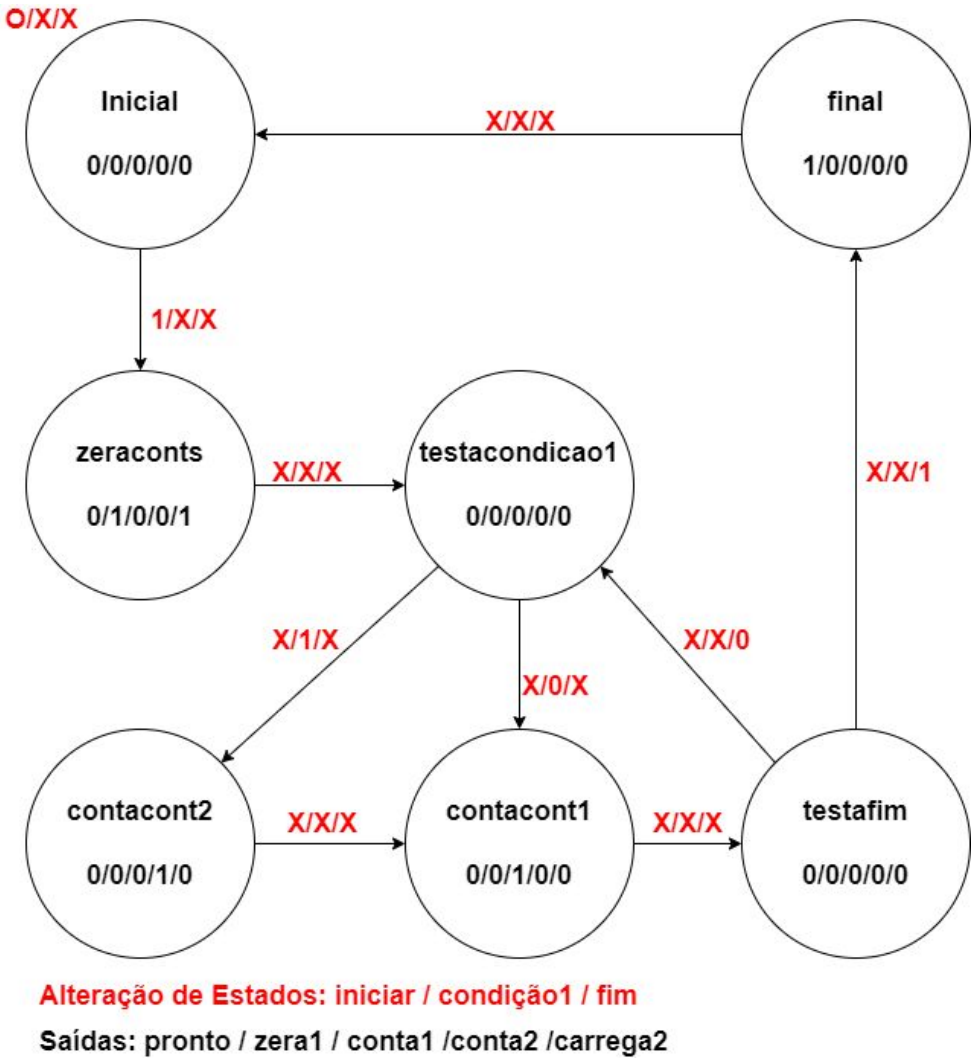


Figura 2: Diagrama de transição da máquina de estados.

Desse diagrama, podemos abstrair que essa máquina de estados tem como objetivo testar uma condição “condicao1” e contar com o contador 2 e em seguida com o contador 1 caso esta seja respeitada ou somente com o contador 1, caso seja desrespeitada. Também podemos observar que esta máquina só inicia e termina seu ciclo de estados quando suas respectivas entradas “iniciar” e “fim” são ativadas.

c. Com este diagrama em mente, tivemos mais facilidade para desenvolver o pseudocódigo:

Pseudocodigo 1-1-c - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Eatual = 'inicial';
db_estado = 000;

pronto = 0;
zera1 = 0;
conta1 = 0;
conta2 = 0;
carrega2 = 0;
db_estado = 0;

while (1)
{
        switch (Eatual)
        {
                case 'inicial'
                {
                        if inciar == 1
                        {
                                estado = 'zeraconts';
                                db_estado = 001;
                                carrega2 = 1;
                                zera1 = 1;
                        }
                        else
                        {
                                estado = 'inicial';
                                db_estado = 000;
                        }
                        break;
                }

                case 'zeraconts'
                {
                        estado = 'testacondicao1';
                        carrega2 = 0;
                        zera1 = 0;
                        db_estado = 010;
                        break;
                }

                case 'testacondicao1'
                {
                        if condicao == 1
                        {
                                estado = 'contacont2'
                                conta2 = 1;
                                db_estado = 011;          }
                                break;
                        }
                        else
                        {
                                estado = 'contacont1'
                                conta1 = 1;
                                break;
                        }
                        break;
                }

                case 'contacont2'
                {
                        estado = 'contacont1'
                        conta2 = 0;
                        conta1 = 1;
                        db_estado = 100;
                        break;
                }

                case 'contacont1'
                {
                        estado = 'testafim'
                        conta1 = 0;
                        db_estado = 101;
                        break;
                }

                case 'testafim'
                {
                        if fim == 1
                        {
                                estado = 'final';
                                pronto = 1;
                                db_estado = 111;
                                break;
                        }
                        else
                        {
                                estado = 'testacondicao1';
                                db_estado = 010;
                                break;
                        }
                }

                case 'final'
                {
                        estado = 'inicial';
                        pronto = 0;
                        db_estado = 000;
                        break;
                }
```

Figura 3: Pseudocódigo equivalente da máquina de estados.

d. A máquina descrita pelo código fornecido é uma máquina de Mealy com 8 estados. Na máquina, há 5 sinais de entrada e 6
sinais de saída.
A máquina pode seguir o caminho descrito no diagrama de transições, de acordo com os sinais de entrada iniciar, condicao1 e fim, ou voltar para o primeiro estado (iniciar) se o sinal reset for 1.
As saídas, zera1, conta1, conta2, carrega2 e db_estado são alteradas de acordo com o estado atual.

e. Simulamos a maquina de estados no Quartus, obtendo os seguintes resultados:

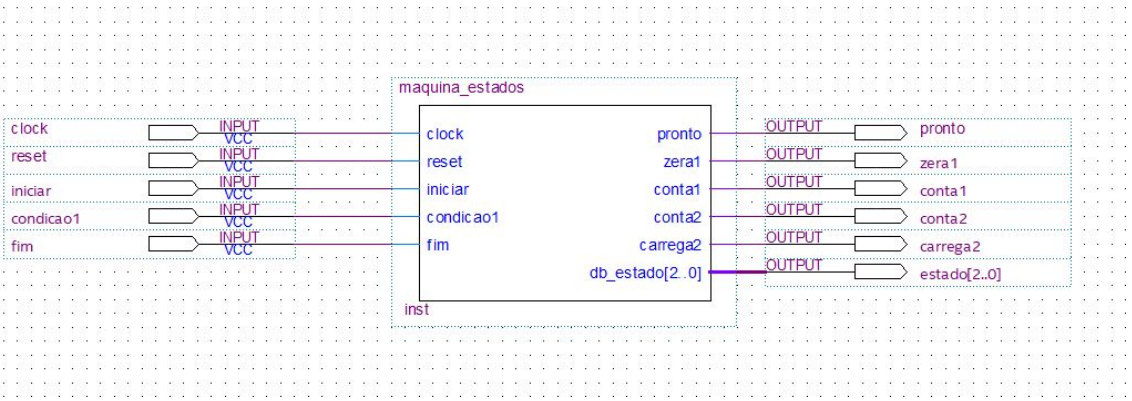


Figura 4: Montagem da Máquina de estados

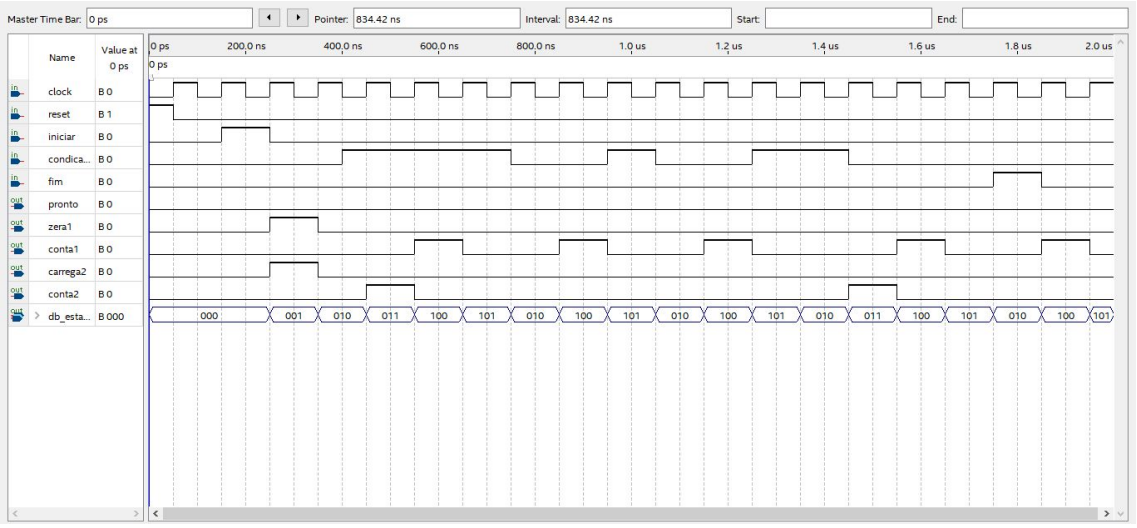


Figura 5: Simulação da máquina de estados

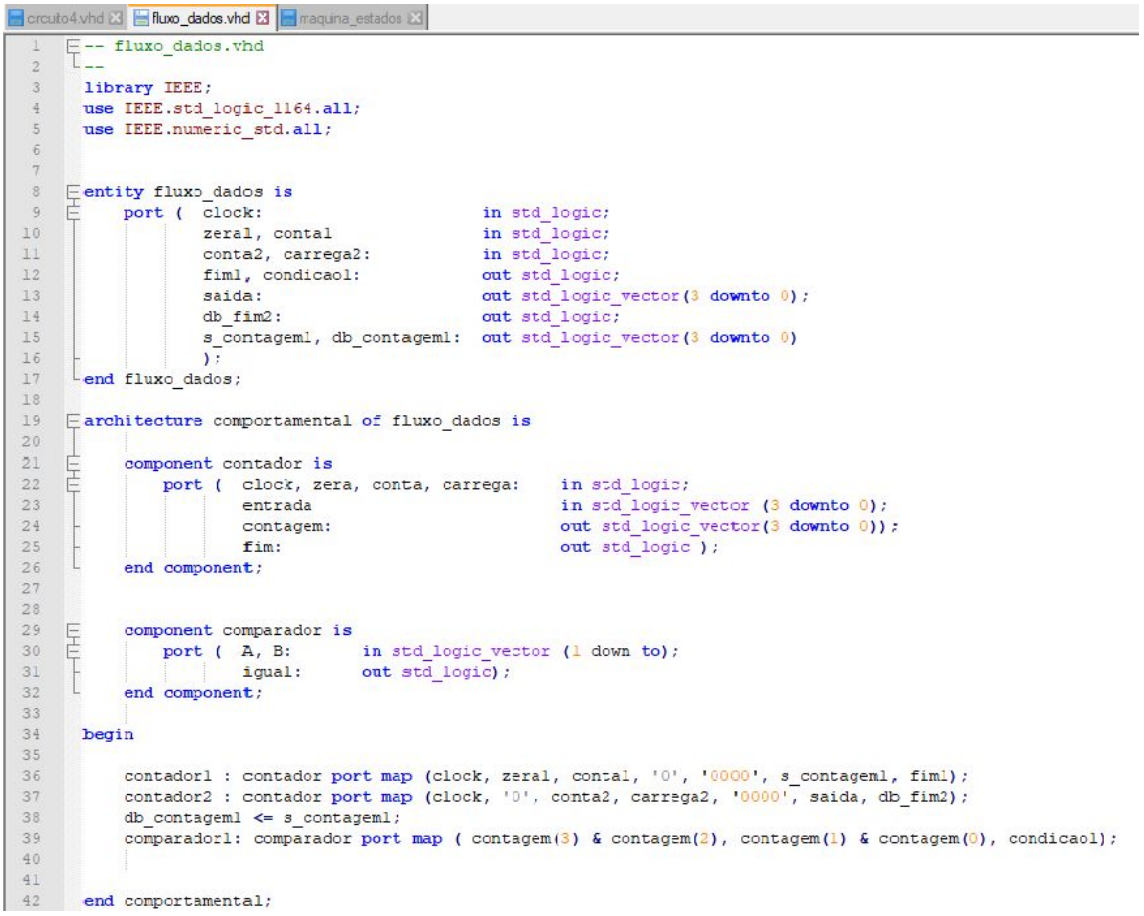
2. Projeto de um Fluxo de Dados em VHDL

a. Modificamos a interface de sinais de o nome do componente de acordo com a definição dada:

```
entity fluxo_dados is
    port (
        clock:          in std_logic;
        zera1, conta1:  in std_logic;
        conta2, carrega2: in std_logic;
        fim1, condicao1:  out std_logic;
        saida:          out std_logic_vector(3 downto 0);
        db_fim2:        out std_logic;
        db_contagem1:   out std_logic_vector(3 downto 0)
    );
end entity;
```

Figura 6: formato da interface externa de sinais do sistema.

b. Com este formato, desenvolvemos o seguinte modelo estrutural interligando os componentes:



```vhdl
-- fluxo_dados.vhd
--
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;


entity fluxo_dados is
    port (  clock:                      in std_logic;
            zera1, conta1               in std_logic;
            conta2, carrega2:           in std_logic;
            fim1, condicao1:            out std_logic;
            saida:                      out std_logic_vector(3 downto 0);
            db_fim2:                    out std_logic;
            s_contagem1, db_contagem1:  out std_logic_vector(3 downto 0)
            );
end fluxo_dados;

architecture comportamental of fluxo_dados is

    component 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 component;


    component comparador is
        port (  A, B:       in std_logic_vector (1 down to);
                igual:      out std_logic);
    end component;

begin

    contador1 : contador port map (clock, zera1, conta1, '0', '0000', s_contagem1, fim1);
    contador2 : contador port map (clock, '0', conta2, carrega2, '0000', saida, db_fim2);
    db_contagem1 <= s_contagem1;
    comparador1: comparador port map ( contagem(3) & contagem(2), contagem(1) & contagem(0), condicao1);


end comportamental;
```

Figura 7: Projeto estrutural do fluxo de dados em VHDL.

c. Dado que o circuito utiliza o contador e o comparador que já havíamos testado e simulado, a montagem deve ser simples.
Testaremos se as entradas zera1 e e carrega2 funcionam como esperado, respectivamente zerando db_contagem1 e carregando “0000” na saída do contador2.

Após estes testes, basta checar se saídas do fluxo de dado tem valor 1 apenas nos momentos esperados.

d. Para melhor avaliar o funcionamento do circuito, o simulamos no Quartus e obtivemos os seguintes resultados:

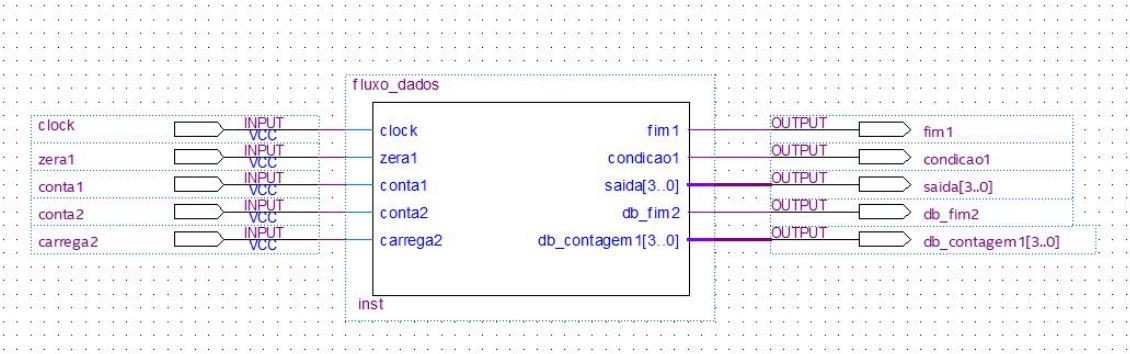


Figura 8: Montagem do circuito fluxo_dados

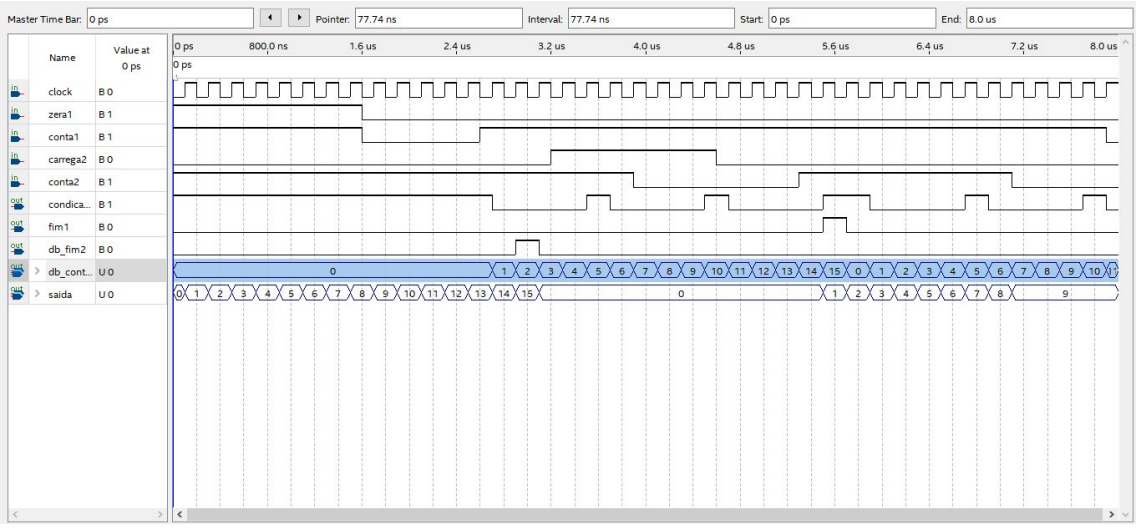


Figura 9: Simulação do fluxo de dados

e. Designação dos pinos na FPGA:

Sinal	Ligação na Placa FPGA	Pino na FPGA
clock	GPIO	N16
zera1	chave SW0	U13
conta1	chave SW1	V13
carrega2	chave SW2	T13
conta2	chave SW3	T12
saída[0]	led LEDR0	AA2
saída[1]	led LEDR1	AA1
saída[2]	led LEDR2	W2
saída[3]	<não designar>	--------

condicao1	led LEDR3	Y3
fim1	led LEDR4	N2
db_contagem[0]	led LEDR5	N1
db_contagem[1]	led LEDR6	U2
db_contagem[2]	led LEDR7	U1
db_contagem[3]	led LEDR8	L2
fim	led LEDR9	L1

Tabela 1: designação de pinos na FPGA.

3. Projeto de um Sistema Digital

a. Devemos projetar um circuito de acordo com as seguintes figuras:

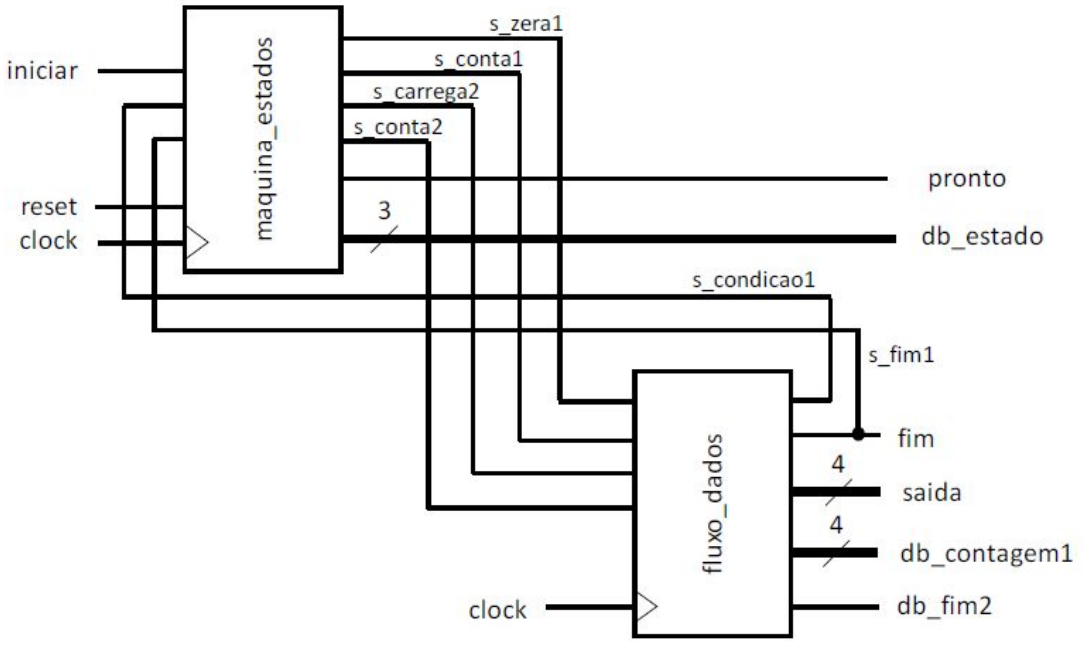


Figura 10: diagrama de blocos circuito 4

```
entity circuito4 is
    port ( clock, reset, iniciar:   in std_logic;
           pronto, fim:             out std_logic;
           saida:                   out std_logic_vector(3 downto 0);
           db_contagem1:            out std_logic_vector(3 downto 0);
           db_zera1, db_conta1:     out std_logic;
           db_conta2, db_carrega2:  out std_logic;
           db_condicao1, db_fim2:   out std_logic;
           db_estado:               out std_logic_vector(2 downto 0)
    );
end circuito4;
```

Figura 11: formato esperado da interface externa de sinais.

Levando em consideração estas imagens, projetamos o seguinte circuito em vhdl:

```vhdl
-- circuito4.vhd
--
library IEEE;
use IEEE.std_logic_1164.all;

entity circuito4 is
    port(   clock, reset, iniciar:  in std_logic;
            pronto, fim:            out std_logic;
            saida:                  out std_logic_vector(3 downto 0);
            db_contagem1:           out std_logic_vector(3 downto 0);
            db_zera1, db_contal:    out std_logic;
            db_conta2, db_carrega2  out std_logic;
            db_condicao1, db_fim2:  out std_logic;
            db_estado:              out std_logic_vector(2 downto 0)
    );
end circuito4;

architecture comportamental of circuito4 is

    component maquina_estados is
        port (  clock, reset:               in std_logic;
                iniciar, condicao1, fim:    in std_logic;
                pronto, zera1, contal:      out std_logic;
                conta2, carrega2:           out std_logic;
                db_estado:                  out std_logic_vector(2 downto 0);
                );
    end component maquina_estados;

    component fluxo_dados is
        port (  clock:                      in std_logic;
                zera1, contal               in std_logic;
                conta2, carrega2:           in std_logic;
                fim1, condicao1:            out std_logic;
                saida:                      out std_logic_vector(3 downto 0);
                db_fim2:                    out std_logic;
                db_contagem1:               out std_logic_vector(3 downto 0)
                );
    end component fluxo_dados;

    maquina : maquina_estados port map(clock, reset, inicia, db_condicao1, fim, pronto, db_zera1, db_contal, db_conta2, db_carrega2, db_estado);
    fluxo   : fluxo_dados port map (clock, db_zera1, db_contal, db_conta2, db_carrega2, fim, db_condicao1, saida, db_fim2, db_contagem1);
end comportamental;
```

Figura 12: circuito 4 em vhdl

b. No momento em que o sinal ‘pronto’ é ativado, a saída apresenta valor 0011 (ou 3 em decimal).
Ao analisar o circuito e observar a simulação (abaixo) foi possível entender por quais passos o circuito teve de passar antes de emitir o sinal ‘pronto’.
Também foi possível perceber que, devido ao fato de que o circuito retorna ao seu estado inicial após atingir o estado ‘final’, o circuito continuará apresentando o valor 3 em sua saída ao emitir o sinal pronto novamente.

c. Realizamos a simulação do funcionamento do circuito 4 e obtivemos o seguinte resultado:

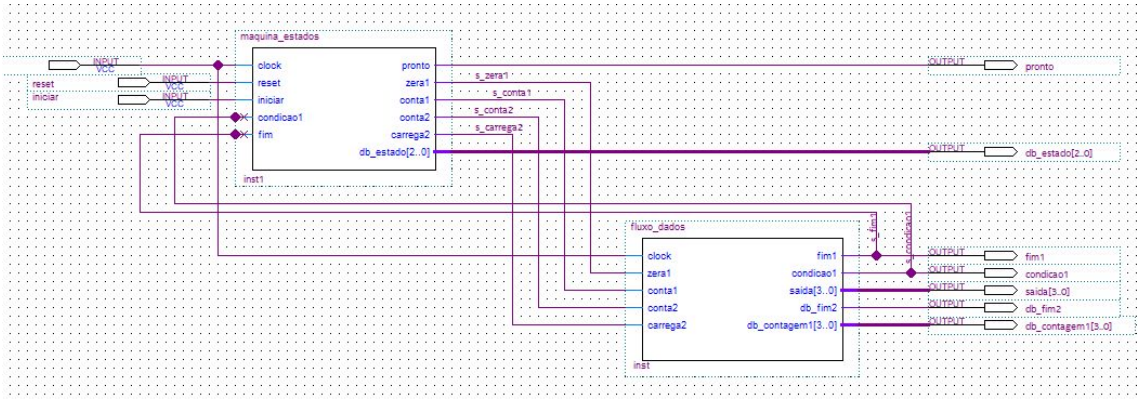


Figura 13: Montagem do Circuito 4 (abrir o projeto para mais detalhes)

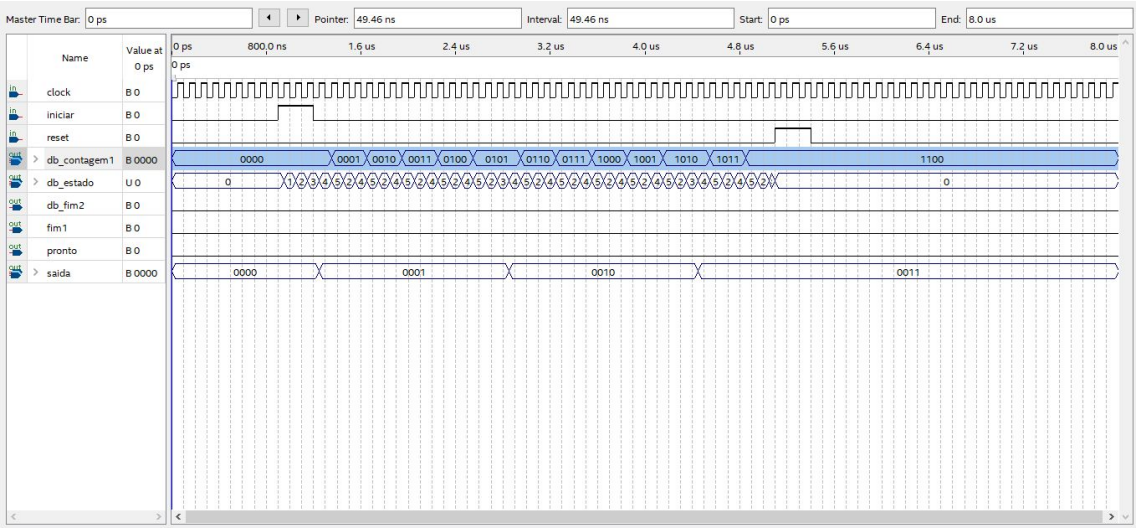


Figura 14: Simulação do Circuito 4

d. Designação dos pinos na FPGA para circuito4:

Sinal	Ligação na Placa FPGA	Pino na FPGA
clock	GPIO	N16
reset	chave SW0	U13
iniciar	chave SW1	V13
saída[0]	led LEDR0	AA2
saída[1]	led LEDR1	AA1
saída[2]	led LEDR2	W2
saída[3]	<não designar>	-------
db_contagem1[0]	led LEDR3	Y3
db_contagem1[1]	led LEDR4	N2
db_contagem1[2]	led LEDR5	N1
db_contagem1[3]	led LEDR6	U2
db_estado[0]	led LEDR7	U1
db_estado[1]	led LEDR8	L2
db_estado[2]	led LEDR9	L1

pronto	HEX 00	U21
fim	HEX 01	V21
db_zera1	HEX 02	W22
db_conta1	HEX 03	W21
db_conta2	HEX 04	Y22
db_carrega2	HEX 05	Y21
db_condicao1	HEX 06	AA22
db_fim2	<não designar>	------

Referências

4. Apostilas e documentos de apoio do site ~/labdig do PCS.
5. Apostilas disponíveis na plataforma e-Disciplinas.