UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE TECNOLOGIAS
CURSO DE ENGENHARIA ELÉTRICA

LUCAS FILUS RAMOS - GRR20202598
VITÓRIA MARIANA DA ROCHA GASINO - GRR20202564

TE351 DA – LABORATÓRIO 5

CURITIBA
2024

UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE TECNOLOGIA
CURSO DE ENGENHARIA ELÉTRICA

Lucas Filus Ramos
Vitória Mariana da Rocha Gasino

TE 351 DA – LABORATÓRIO 5

Relatório técnico apresentado à disciplina Microeletrônica do Curso de Engenharia Elétrica do Setor de Tecnologia da Universidade Federal do Paraná, como parcial para completamento da disciplina.

Orientador: Sibilla Batista da Luz Franca.

CURITIBA

2024

Sumário

1. INTRODUÇÃO

O objetivo deste projeto consiste na realização de um controlador de portão de garagem em VHDL, utilizando o conceito de máquina de estados.

2. PROJETO 1 – Controle de Portão de Garagem

Neste projeto, foi realizado cinco processos diferentes para que todas as especificações fossem atendidas.

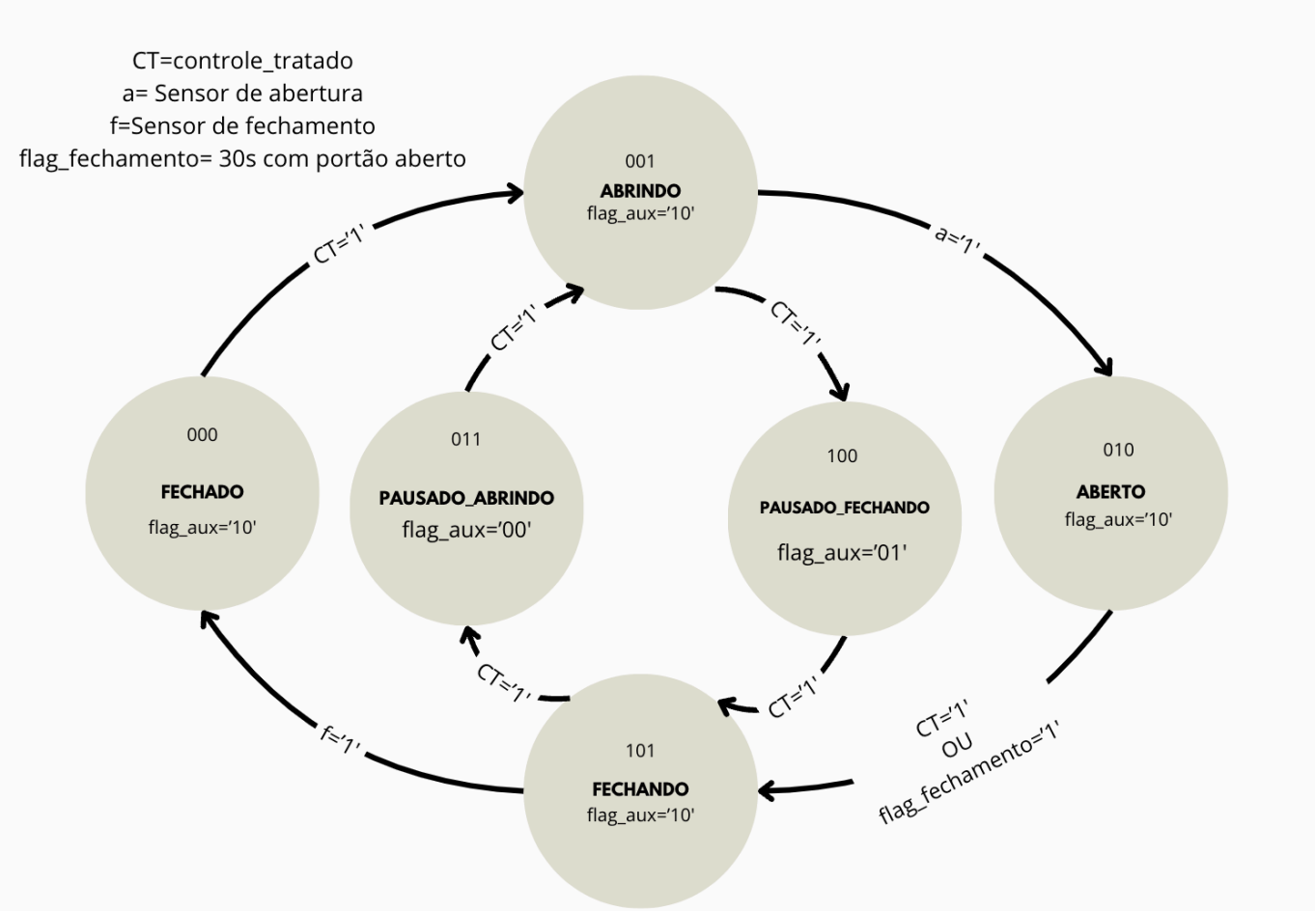
Um processo para o debounce do botão(entrada '*controle'*), um processo para voltar a flag de debounce(*controle_tratado*) para '0', um processo para fazer a mudança dos estados da máquina, um processo para a contagem de 30s caso o portão esteja no estado *'abrindo'* , um processo para a máquina de estados.

Para as saídas, foram atribuídos dois Leds da placa, onde: se 'ligar' estiver no estado alto, significa que o portão está ligado; e, em relação à 'direcao'', se '1' significa que o portão está abrindo, e se '0' significa que o portão está fechando.

O diagrama a seguir apersenta os estados da máquina definidos:

FIGURA 1 – Diagrama de transição.

FONTE: Os autores



Como foram definidos seis estados, temos que o número de flip-flops utilizados para esta máquina é igual a três.

É importante destacar que foi definido como estado inicial o estado 'fechando', onde quando ativado o 'rst, a máquina retorna para o estado '101', e a saída esperad é 'ligar'='1'e 'direcao'='0';

2.1. Desenvolvimento do código

As figuras a seguir ilustram a entidade top e códigos de cada componente.

FIGURA 3 – Código da entidade top.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4
5  entity lab_5 is
6        GENERIC( clock_placa: positive :=100;
7                 debounce_valor: positive := 10); --50 MHz
8        PORT ( clk,rst, abt, fch, controle : in std_logic;
9               ligar, direcao: out std_logic);
10 end lab_5;
11
12 architecture Behavioral of lab_5 is
13    type estado is(fechado,abrindo,aberto,pausado_fechando,pausado_abrindo,fechando);
14    signal pr_state, nx_state : estado;
15    signal flag_fechamento, controle_tratado, reset_flag: std_logic :='0';
16
17    COMPONENT debounce is
18          GENERIC( default_delay: positive := 50000);
19          PORT( clk, bt, reset : in std_logic;
20                y: out std_logic);
21    END COMPONENT;
22
23 begin
24
25    debounce_bt: debounce GENERIC MAP(debounce_valor) PORT MAP (clk,controle,reset_flag,controle_tratado);
26
27
28    process(rst,clk)
29    begin
30       if(rst='1') then
31          pr_state<=fechando; -- Perguntas
32       elsif(clk'event and clk='1') then
33          pr_state<= nx_state;
34       end if;
35    end process;
36
37    process(clk,rst)
38    variable cont: integer :=0;
39    begin
40       if(clk'event and clk='1') then
41          if(abt='1') then
42             cont:=cont+1;
43             if(cont=clock_placa*30) then
44                cont:=0;
45                flag_fechamento<='1';
46             end if;
47          elsif(pr_state = fechado OR pr_state= fechando) then
48             flag_fechamento <='0';
49          end if;
50       end if;
51    end process;
```

```
53    process(pr_state, controle_tratado , abt, fch,flag_fechamento)
54    begin
55       case pr_state is
56          when fechado =>
57             ligar<='0';
58             direcao<='0';
59             if (controle_tratado='1') then
60                reset_flag<='1';
61                nx_state<=abrindo;
62             else
63                reset_flag<='0';
64                nx_state<=fechado;
65             end if ;
66          when abrindo =>
67             ligar<='1';
68             direcao<='1'; -- 1 significa abrindo
69             if( abt='1') then
70                nx_state<= aberto;
71                reset_flag<='0';
72             elsif(controle_tratado='1') then
73                reset_flag<='1';
74                nx_state<= pausado_fechando;
75             else
76                nx_state<=abrindo;
77                reset_flag<='0';
78             end if;
79          when aberto =>
80             ligar<='0';
81             direcao<='0';
82             if (controle_tratado='1' OR flag_fechamento='1') then
83                --flag_fechamento<='0';
84                reset_flag<='1';
85                nx_state<=fechando;
86             else
87                nx_state<=aberto;
88                reset_flag<='0';
89             end if;

90          when fechando =>
91             ligar<='1';
92             direcao<='0';
93             if (controle_tratado='1') then
94                nx_state<= pausado_abrindo;
95                reset_flag<='1';
96                --flag_aux:="00"; --> fechando-->pausa-->ab rindo
97             elsif ( fch='1') then
98                nx_state<=fechado;
99                reset_flag<='0';
100            else
101               nx_state<=fechando;
102               reset_flag<='0';
103            end if;
104
105         when pausado_fechando =>
106            ligar<='0';
107            direcao<='0';
108            if(controle_tratado='1') then
109                  nx_state<= fechando;
110                  reset_flag<='1';
111
112            else
113                  nx_state<= pausado_fechando;
114                  reset_flag<='0';
115            end if;
116         when pausado_abrindo =>
117            ligar<='0';
118            direcao<='0';
119            if(controle_tratado='1') then
120                  nx_state<= abrindo;
121                  reset_flag<='1';
122
123            else
124                  nx_state<= pausado_abrindo;
125                  reset_flag<='0';
126            end if;
127
128
129         when others =>
130            ligar<='0';
131            direcao<='0';
132            nx_state<=fechando;
133            reset_flag<='0';
134       end case;
135    end process;
136
137 end Behavioral;
```

FONTE: Os autores (2024)

FIGURA 4 – Código do componente “debounce”.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity debounce is
5     GENERIC( default_delay: positive := 50000);
6     PORT( clk, bt, reset: in std_logic;
7           y: out std_logic);
8  end debounce;
9
10 architecture Behavioral of debounce is
11 signal flag: std_logic:='0';
12 begin
13
14    process(clk)
15       variable cont_d:integer range 0 to 50000;
16    begin
17       if(clk'event and clk='1') then
18          if(reset='1') then
19             flag<='0';
20          end if;
21          if(flag/=bt) then
22             cont_d:=cont_d+1;
23             if(cont_d=default_delay) then
24                cont_d:=0;
25                flag<=bt;
26             end if;
27          else
28                cont_d:=0;
29
30          end if;
31       end if;
32       y<=flag;
33    end process;
34
35 end Behavioral;
```

FONTE: Os autores (2024)

2.2. Implementação do Test Bench e resultados

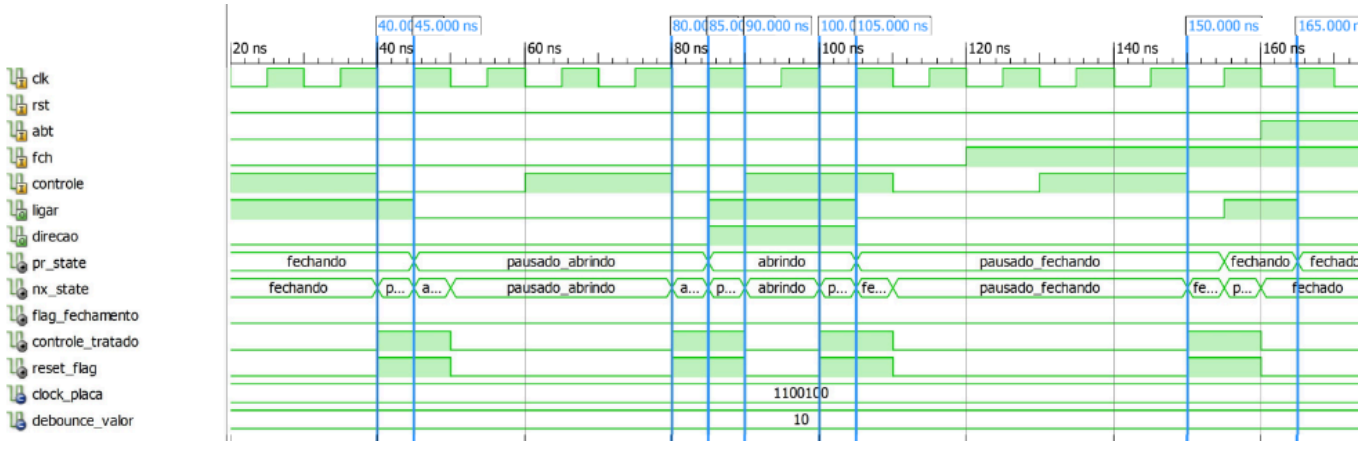
Para avaliar o funcionamento do circuito implementado acima, foi preparado algumas simulações utilizando *clock* em 10 ns. A simulações ilustradas nas a seguir mostram os resultados do test bench.

FIGURA 5 – Código Test Bench.

```
65    -- Stimulus process
66    stim_proc: process
67    begin
68
69       rst<='1';  --fechando
70       wait for clk_period*5;
71       rst<='0';
72       wait for clk_period*5;
73       controle<='1';
74       wait for clk_period*10;
75       controle<='0'; --abrindo
76       wait for clk_period*5;
77       controle<='1';
78       wait for clk_period*10;
79       controle<='0'; --pausado_fechando
80       wait for clk_period*5;
81       controle<='1';
82       wait for clk_period*10;
83       controle<='0'; --fechando
84
85       wait for clk_period*5;
86       fch <='1'; --fechado
87       wait for clk_period*5;
88       controle<='1';
89       wait for clk_period*10;
90       controle<='0'; --abrindo
91       wait for clk_period*5;
92       abt<='1'; --abeto
93       wait for clk_period*10;
94       controle<='1';
95       wait for clk_period*10;
96       controle<='0'; --pausado
97
98
99       wait;
100
101
102
103      wait;
104   end process;
```

FONTE: Os autores (2024).

FIGURA 6 – Simulação do funcionamento 1.

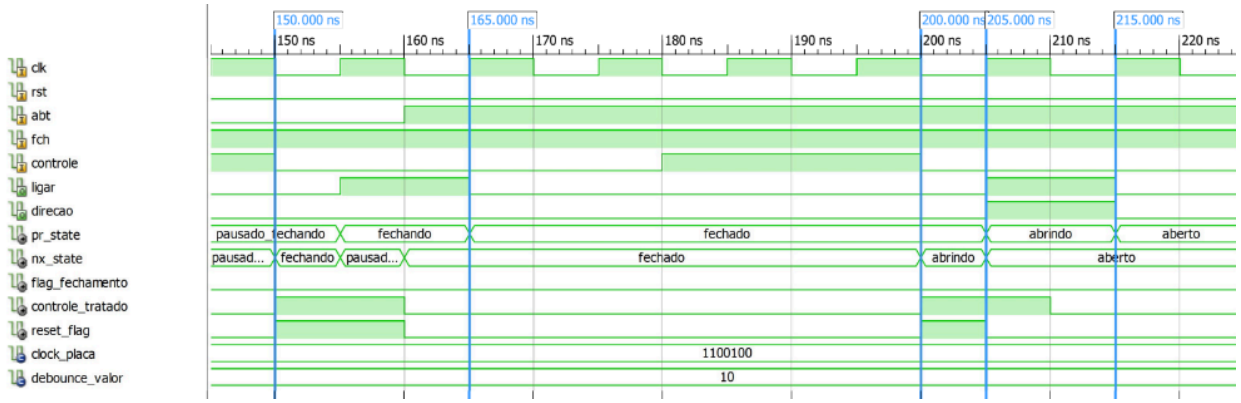


FONTE: Os autores (2024).

FIGURA 7 – Simulação do funcionamento 2.

Para iniciar o teste, foi colocado o 'rst'em nível alto. Com o estado atual em 'fechando', foi simulado o aperto do botão por mais de dois ciclos de clock(debounce definido na entidade top como dois ciclos). Com a subida de 'controle_tratado' em 40 ns, o 'nx_state' automaticamente muda de estado para 'pausado_abrindo'. N apróxima batida, o estado atual('pr_state') assume o 'nx_state'(45 ns).

Na imagem 6, é possível observar que a cada subida de 'controle', caso essa entrada permaneça por 2 ciclos ou mais em alto, sobre 'controle_tratado' que ativa a mudança de estado na máquina



FONTE: Os autores (2024).

Na imagem 7 é possível observar que em 165 ns o estado atual esta em 'fechando'. Quando a entrada 'fch' assume '1', o estado muda assim como o esperado descrito na imagem 1. O mesmo ocorre para amudança de 'abrindo' para 'aberto', onde aos 215 ns da simulação, o estado abrindo muda para 'aberto' um ciclo depois de permanecer em 'abrindo', pois 'abt' está em alto.

O código porém, não funcionou corretamente ao ser executado na placa física. Ao apertar o botão, os estados não são trocados como esperado.

3. CONCLUSÃO

Este relatório apresentou uma implementação de um projeto de controle de portão utilizando FPGA e conceitos básicos de eletrônica digital, como máquina de estado. Apesar dos resultados corretos obtidos na simulação, não foi possível obter os mesmos resultados na implementação física na placa. Além disso, não foi possível identificar a origem do erro.