UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Augusto Cavalcante Barbosa Pereira - 14651531
Ayrton da Costa Ganem Filho - 14560190
Felipe Volkweis de Oliveira - 14570041

Relatório do Trabalho de Máquinas de Estado - Sistemas Digitais

Professor Danilo H. Spatti

São Carlos, SP
11/12/2023

SUMÁRIO

1 INTRODUÇÃO

O trabalho tem como objetivo aplicar os conhecimentos adquiridos na disciplina teórica de Sistemas Digitais dentro do tópico de máquinas de estado.
A seguir será discutida a implementação de duas máquinas de estados distintas:

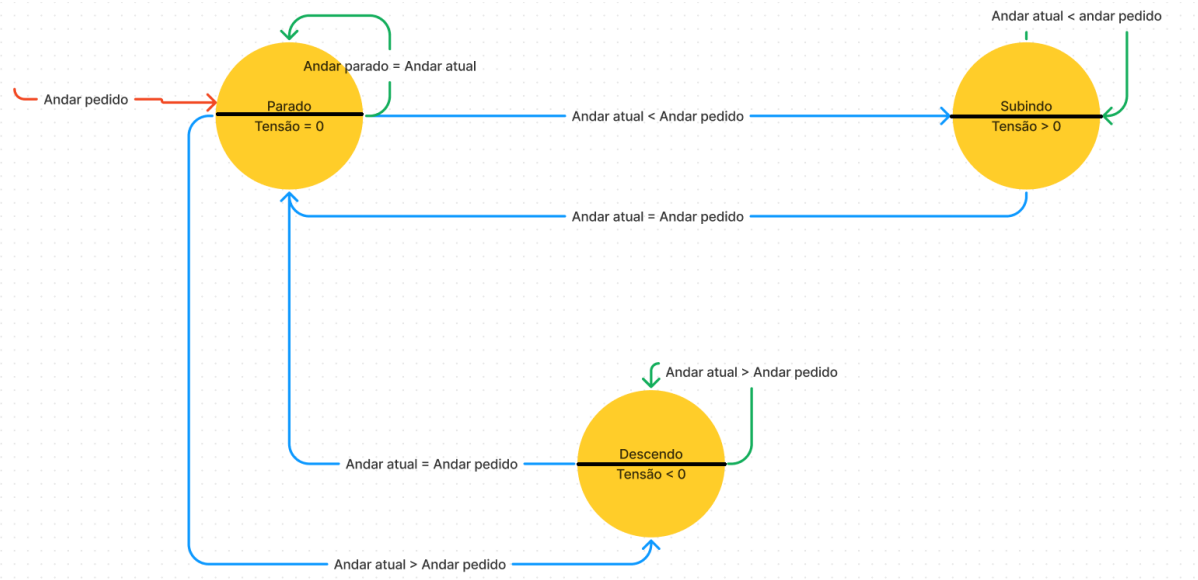
1. um elevador utilizando o modelo de Moore;
2. uma máquina de refrigerante utilizando o modelo de Mealy.

Ambas as máquinas foram implementadas utilizando VHDL e transformadas em blocos esquemáticos dentro do Quartus.

2 ELEVADOR

Implementamos a máquina de estados do elevador utilizando uma máquina de Moore, utilizando uma variável para armazenar o andar atual.

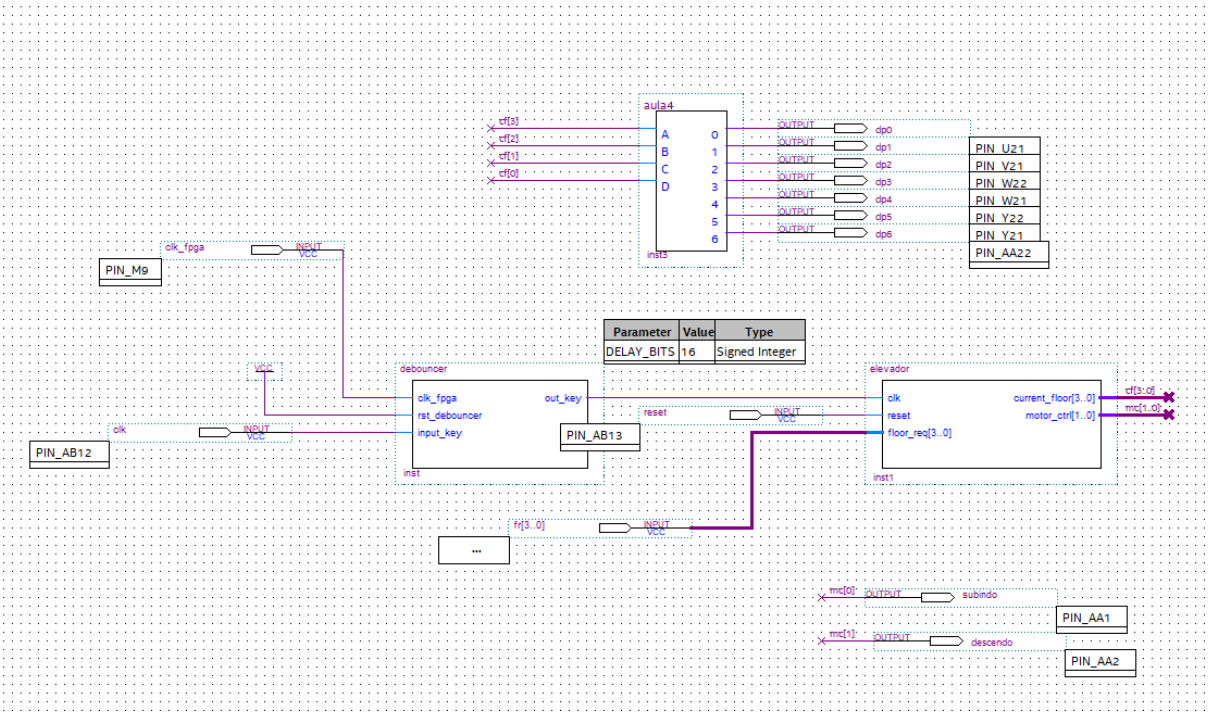
2.1 Diagrama de estados



2.2 Tabela de transição de estados

Estado Atual	Andar Atual > Andar Pedido	Andar Atual = Andar Pedido	Andar Atual < Andar Pedido	Saída
Parado	Descendo	Parado	Subindo	Tensão = 0 (00)
Subindo	—	Parado	Subindo	Tensão > 0 (01)
Descendo	Descendo	Parado	—	Tensão < 0 (10)

2.3 Implementação no Quartus



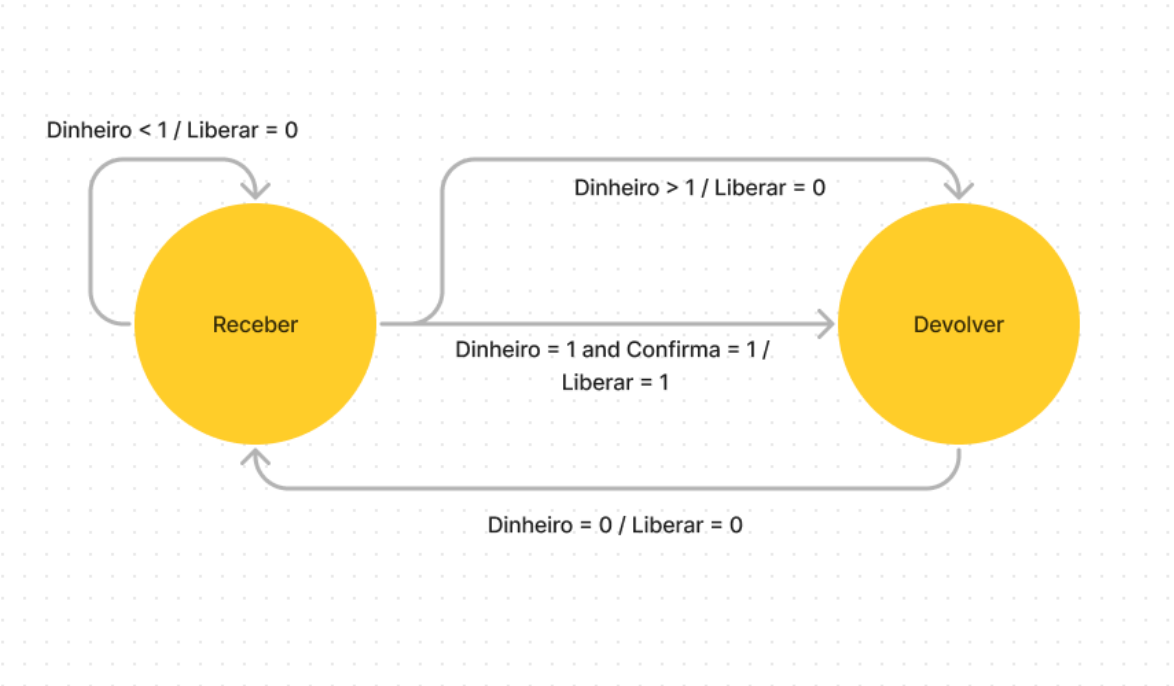
2.4 Código em VHDL

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4
5  entity elevador is
6      Port (
7          clk         : in  STD_LOGIC;
8          reset       : in  STD_LOGIC;
9          floor_req   : in  STD_LOGIC_VECTOR(3 downto 0);
10         current_floor : out STD_LOGIC_VECTOR(3 downto 0);
11         motor_ctrl : out STD_LOGIC_VECTOR(1 downto 0)
12     );
13 end elevador;
14
15 architecture Behavioral of elevador is
16     type state_type is (PARADO, SUBINDO, DESCENDO);
17     signal current_state: state_type := PARADO;
18    begin
19       process(clk,reset)
20       variable andar :STD_LOGIC_VECTOR(3 downto 0) := "0000";
21       begin
22       if(reset = '1') then
23          current_state <= PARADO;
24          andar := "0000";
25       elsif (clk'event) and (clk = '1') then
26          case current_state is
27             WHEN PARADO =>
28                motor_ctrl <= "00";
29                if(floor_req > andar) then
30                   current_state <= SUBINDO;
31                elsif (floor_req < andar) then
32                   current_state <= DESCENDO;
33                elsif (floor_req = andar) then
34                   current_state <= PARADO;
35                end if;
36
37             WHEN SUBINDO =>
38                motor_ctrl <= "01";
39                andar := std_logic_vector(unsigned(andar)+1);
40                if(floor_req > andar) then
41                   current_state <= SUBINDO;
42                elsif (floor_req = andar) then
43                   current_state <= PARADO;
44                end if;
45
46             WHEN DESCENDO =>
47                motor_ctrl <= "10";
48                andar := std_logic_vector(unsigned(andar)-1);
49                if (floor_req < andar) then
50                   current_state <= DESCENDO;
51                elsif (floor_req = andar) then
52                   current_state <= PARADO;
53                end if;
54          end case;
55       current_floor <= andar;
56       end if;
57    end process;
58 end Behavioral;
```

3 MÁQUINA DE REFRIGERANTE

Implementamos a máquina de estados da máquina de refrigerante utilizando o modelo de Mealy, utilizando uma variável para acumular o dinheiro armazenado pela máquina..

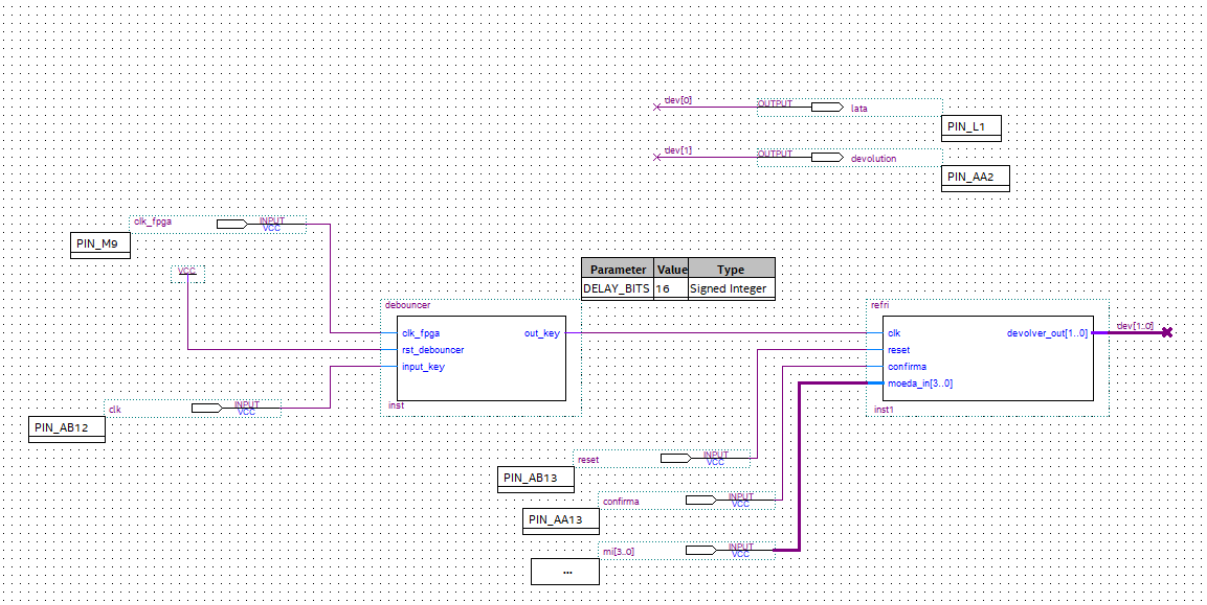
3.1 Diagrama de estados



3.2 Tabela de transição de estados

Estado Atual	Dinheiro < 100	Dinheiro > 100	Dinheiro = 0	Dinheiro = 100 and Confirma = 1
Receber	Receber/ Troco=0 Lata=0	Devolver/ Troco=1 Lata = 0	—	Devolver/ Troco = 0 Lata=1
Devolver	—	—	Receber/ Troco=0 Lata=0	—

3.3 Implementação no Quartus



3.4 Código em VHDL

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4
5  entity refri is
6      Port (
7          clk         : in  STD_LOGIC;
8          reset       : in  STD_LOGIC;
9          confirma      : in  STD_LOGIC;
10         moeda_in  : in  STD_LOGIC_VECTOR(3 downto 0);
11         devolver_out : out STD_LOGIC_VECTOR(1 downto 0)
12     );
13 end refri;
14
15 architecture Behavioral of refri is
16     type state_type is (recebendo, devolvendo);
17     signal current_state: state_type := recebendo;
18 begin
19    process(clk,reset)
20    variable dinheiro :integer:= 0;
21    variable futuro :integer:= 0;
22    begin
23       if(reset = '1') then
24             current_state <= recebendo;
25             dinheiro := 0;
26       elsif (clk'event) and (clk = '1') then
27          case current_state is
28             WHEN recebendo =>
29                if(moeda_in="0001")then
30                   futuro:=10;
31                elsif(moeda_in="0010")then
32                   futuro:=25;
33                elsif(moeda_in="0100")then
34                   futuro:=50;
35                elsif(moeda_in="1000")then
36                   futuro:=100;
37                elsif(moeda_in="0000")then
38                   futuro:=0;
39                end if;
40                dinheiro := dinheiro + futuro;
41                if(dinheiro=100 and confirma ='1') then
42                   futuro:=0;
43                   devolver_out<="01";
44                   current_state<=devolvendo;
45                elsif(dinheiro>100) then
46                   futuro:=0;
47                   devolver_out<="10";
48                   current_state<=devolvendo;
49                end if;
50
51             WHEN devolvendo =>
52                devolver_out <= "00";
53                dinheiro:=0;
54                current_state <= recebendo;
55
56
57          end case;
58       end if;
59    end process;
60 end Behavioral;
```