Universidade de Brasília
Departamento de Engenharia Elétrica

Aluna: Mariana Soares Oliveira
Matrícula: 231013663
Turma 01
18/01/2024

Relatório Experimento 7

1. Introdução

O intuito do seguinte experimento é desenvolver uma máquina de estado síncrona do tipo Moore para controlar uma máquina de refrigerantes, utilizando a linguagem de descrição de hardware VHDL, e simular o seu comportamento por meio de um *testbench* realizado no *software* ModelSim.

2. Teoria

2.1 Máquinas de Estado: tipo Moore e Mealy

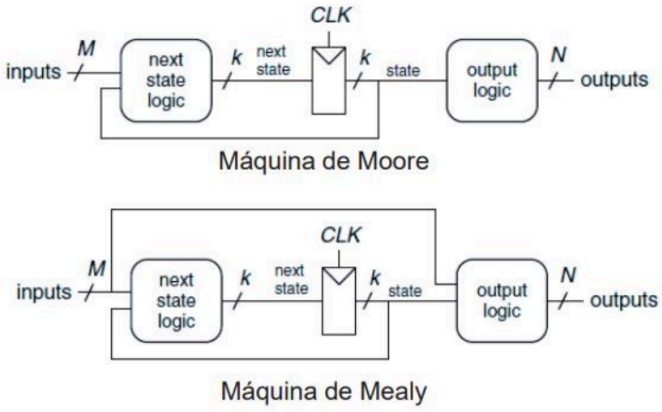

Figura 1. Estrutura da Máquina de Moore e Mealy

As máquinas de estado finito são modelos usados em sistemas digitais, onde o comportamento é determinado por entradas, estados e saídas. Existem dois tipos principais: Máquina de Moore e Máquina de Mealy, que diferem em como geram suas saídas.

Na máquina de Moore, as saídas dependem exclusivamente do estado atual. Isso significa que, para um dado estado, a saída será sempre a mesma, independentemente do valor das entradas. Essa característica garante maior estabilidade, pois a saída só muda quando há uma transição de estado. Na Figura 1, podemos observar que, na máquina de Moore, a lógica de saída (output logic) está conectada diretamente ao estado armazenado (state). Assim, as entradas não influenciam diretamente as saídas, apenas definem a próxima transição de estado.

Já na máquina de Mealy, as saídas dependem tanto do estado atual quanto das entradas. Essa arquitetura permite que o sistema responda mais rapidamente a alterações nas entradas, já que as saídas podem mudar sem que ocorra uma transição de estado. Na Figura 1, a máquina de Mealy mostra que a lógica de saída (output logic) recebe tanto o estado atual (state) quanto as entradas (inputs). Essa conexão direta torna as máquinas de Mealy mais dinâmicas, mas também mais sensíveis a variações nas entradas.

2.2. Máquina de Refrigerante

Esse experimento consiste na implementação de uma máquina de estados do tipo Moore para controlar uma máquina de refrigerantes que aceita moedas de R$ 0,25 e R$ 0,50. A máquina soma os valores inseridos e, ao atingir ou exceder R$ 1,00, libera automaticamente o refrigerante e o troco, se necessário. Ela também permite o cancelamento da compra, devolvendo o valor inserido. A lógica segue a Tabela 1, que define os estados, transições e saídas.

Estados	Saídas			Entradas			
	R	0,25 C	0,50 C	A = 00	A = 01	A = 10	A = 11
INIT	0	0	0	INIT	e0,25	e0,50	INIT
e0,25	0	0	0	e0,25	e0,50	e0,75	d0,25
e0,50	0	0	0	e0,50	e0,75	e1,00	d0,50
e0,75	0	0	0	e0,75	e1,00	e1,25	d0,75
e1,00	1	0	0	INIT	e0,25	e0,50	INIT
e1,25	1	1	0	INIT	e0,25	e0,50	INIT
d0,25	0	1	0	INIT	e0,25	e0,50	INIT
d0,50	0	0	1	INIT	e0,25	e0,50	INIT
d0,75	0	1	1	INIT	e0,25	e0,50	INIT

Tabela 1. Tabela de transição de estados da Máquina de Refrigerante

3. Códigos

Neste experimento utilizamos a linguagem de descrição de hardware VHDL por meio do software Modelsim para desenvolver uma máquina de estados conforme a figura 2. Posteriormente, foi desenvolvido um código auxiliar chamado *testbench* para cada circuito, descrito na figura 3.

C:/Users/maria/Desktop/231013663_Projeto7/questao1/maquinarefri.vhd - Default

```
Ln#
 1  -- biblioteca
 2  library IEEE;
 3  use IEEE.STD_LOGIC_1164.ALL;
 4  -- entidade
 5  entity maquinarefri is
 6          port (
 7                  clk :in std_logic;
 8                  moeda :in std_logic_vector(1 downto 0);
 9                  r, c25, c50 :out std_logic
10  );
11  end maquinarefri;
12  -- arquitetura
13  architecture rtl of maquinarefri is
14          type state is (idle, e25c, e50c, e75c, e1, e125, d25c, d50c, d75c);
15          signal currentstate, nextstate : state;
16  begin
17  sync_proc: process(clk)
18          begin
19                  if rising_edge(clk) then
20                          currentstate <= nextstate;
21                  end if;
22  end process;
23  comb_proc: process (currentstate, moeda)
24  begin
25          case currentstate is
26                  when idle =>
27                          r <= '0';
28                          c25 <= '0';
29                          c50 <= '0';
30                          if (moeda = "01") then nextstate <= e25c;
31                          elsif (moeda = "10") then nextstate <= e50c;
32                          elsif (moeda = "11") then nextstate <= idle;
33                          else nextstate  <= idle;
34                          end if;
35                  when e25c =>
36                          r <= '0';
37                          c25 <= '0';
38                          c50 <= '0';
39                          if (moeda = "01") then nextstate <= e50c;
40                          elsif (moeda = "10") then nextstate <= e75c;
41                          elsif (moeda = "11") then nextstate <= d25c;
42                          else nextstate  <= e25c;
43                          end if;
44                  when e50c =>
```

```
45                      r <= '0';
46                      c25 <= '0';
47                      c50 <= '0';
48                      if (moeda = "01") then nextstate <= e75c;
49                      elsif (moeda = "10") then nextstate <= e1;
50                      elsif (moeda = "11") then nextstate <= d50c;
51                      else nextstate  <= e50c;
52                      end if;
53              when e75c =>
54                      r <= '0';
55                      c25 <= '0';
56                      c50 <= '0';
57                      if (moeda = "01") then nextstate <= e1;
58                      elsif (moeda = "10") then nextstate <= e125;
59                      elsif (moeda = "11") then nextstate <= d75c;
60                      else nextstate  <= e75c;
61                      end if;
62              when e1 =>
63                      r <= '1';
64                      c25 <= '0';
65                      c50 <= '0';
66                      if (moeda = "01") then nextstate <= e125;
67                      elsif (moeda = "10") then nextstate <= e50c;
68                      elsif (moeda = "11") then nextstate <= idle;
69                      else nextstate  <= idle;
70                      end if;
71              when e125 =>
72                      r <= '1';
73                      c25 <= '1';
74                      c50 <= '0';
75                      if (moeda = "01") then nextstate <= e25c;
76                      elsif (moeda = "10") then nextstate <= e50c;
77                      elsif (moeda = "11") then nextstate <= idle;
78                      else nextstate  <= idle;
79                      end if;
80              when d25c =>
81                      r <= '0';
82                      c25 <= '1';
83                      c50 <= '0';
84                      if (moeda = "01") then nextstate <= e25c;
85                      elsif (moeda = "10") then nextstate <= e50c;
86                      elsif (moeda = "11") then nextstate <= idle;
87                      else nextstate  <= idle;
88                      end if;
```

```
89                  when d50c =>
90                          r <= '0';
91                          c25 <= '0';
92                          c50 <= '1';
93                          if (moeda = "01") then nextstate <= e25c;
94                          elsif (moeda = "10") then nextstate <= e50c;
95                          elsif (moeda = "11") then nextstate <= idle;
96                          else nextstate  <= idle;
97                          end if;
98                  when d75c =>
99                          r <= '0';
100                         c25 <= '1';
101                         c50 <= '1';
102                         if (moeda = "01") then nextstate <= e25c;
103                         elsif (moeda = "10") then nextstate <= e50c;
104                         elsif (moeda = "11") then nextstate <= idle;
105                         else nextstate  <= idle;
106                         end if;
107
108                 end case;
109         end process;
110 end rtl;
```

Figura 2. Codificação da questão 1

C:/Users/maria/Desktop/231013663_Projeto7/questao1/tb_maquinarefri.vhd (/tb_maquinarefri) - Default

```
Ln#
  1  -- biblioteca
  2  library IEEE;
  3  use IEEE.STD_LOGIC_1164.ALL;
  4  use IEEE.NUMERIC_STD.ALL;
  5  -- entidade
  6  entity tb_maquinarefri is
  7  end tb_maquinarefri;
  8  -- arquitetura
  9  architecture testbench of tb_maquinarefri is
 10      -- sinais do testbench
 11      signal clk : std_logic := '0';
 12      signal moeda : std_logic_vector(1 downto 0) := "00";
 13      signal r : std_logic;
 14      signal c25 : std_logic;
 15      signal c50 : std_logic;
 16      constant clk_period : time := 10 ns;
 17      -- componente maquinarefri
 18      component maquinarefri is
 19          port (
 20              clk : in std_logic;
 21              moeda : in std_logic_vector(1 downto 0);
 22              r : out std_logic;
 23              c25 : out std_logic;
 24              c50 : out std_logic
 25          );
 26      end component;
 27  begin
 28      -- instancia
 29      uut: maquinarefri
 30      port map (
 31          clk => clk,
 32          moeda => moeda,
 33          r => r,
 34          c25 => c25,
 35          c50 => c50
 36      );
 37      -- processo para gerar o clock
 38      clk_process: process
 39      begin
 40          while true loop
 41              clk <= '0';
 42              wait for clk_period / 2;
 43              clk <= '1';
 44              wait for clk_period / 2;
```

```
45          end loop;
46      end process;
47
48      stim_proc: process
49          type input_array is array (0 to 3) of std_logic_vector(1 downto 0);
50          constant inputs : input_array := ("00", "01", "10", "11");
51      begin
52
53          for i in inputs'range loop
54              moeda <= inputs(i); wait for clk_period;
55          end loop;
56
57          moeda <= "01"; wait for clk_period;
58          for i in inputs'range loop
59              moeda <= inputs(i); wait for clk_period;
60          end loop;
61
62          moeda <= "10"; wait for clk_period;
63          for i in inputs'range loop
64              moeda <= inputs(i); wait for clk_period;
65          end loop;
66
67          moeda <= "01"; wait for clk_period;
68          for i in inputs'range loop
69              moeda <= inputs(i); wait for clk_period;
70          end loop;
71
72          moeda <= "01"; wait for clk_period;
73          for i in inputs'range loop
74              moeda <= inputs(i); wait for clk_period;
75          end loop;
76
77          moeda <= "01"; wait for clk_period;
78          for i in inputs'range loop
79              moeda <= inputs(i); wait for clk_period;
80          end loop;
81
82          moeda <= "11"; wait for clk_period;
83          for i in inputs'range loop
84              moeda <= inputs(i); wait for clk_period;
85          end loop;
86
87          moeda <= "11"; wait for clk_period;
88          for i in inputs'range loop
89              moeda <= inputs(i); wait for clk_period;
90          end loop;
91
92          moeda <= "11"; wait for clk_period;
93          for i in inputs'range loop
94              moeda <= inputs(i); wait for clk_period;
95          end loop;
96
97          moeda <= "01"; wait for clk_period;
98          moeda <= "11"; wait for clk_period;
99          for i in inputs'range loop
100             moeda <= inputs(i); wait for clk_period;
101         end loop;
102
103         moeda <= "10"; wait for clk_period;
104         moeda <= "11"; wait for clk_period;
105         for i in inputs'range loop
106             moeda <= inputs(i); wait for clk_period;
107         end loop;
108         wait;
109     end process;
110
111 end testbench;
```

Figura 3. *Testbench* da questão 1

4. Compilação

Os códigos gerados anteriormente foram submetidos a uma compilação com o intuito de garantir seu funcionamento, como mostrado na figura 4, não apresentando erros de sintaxe.

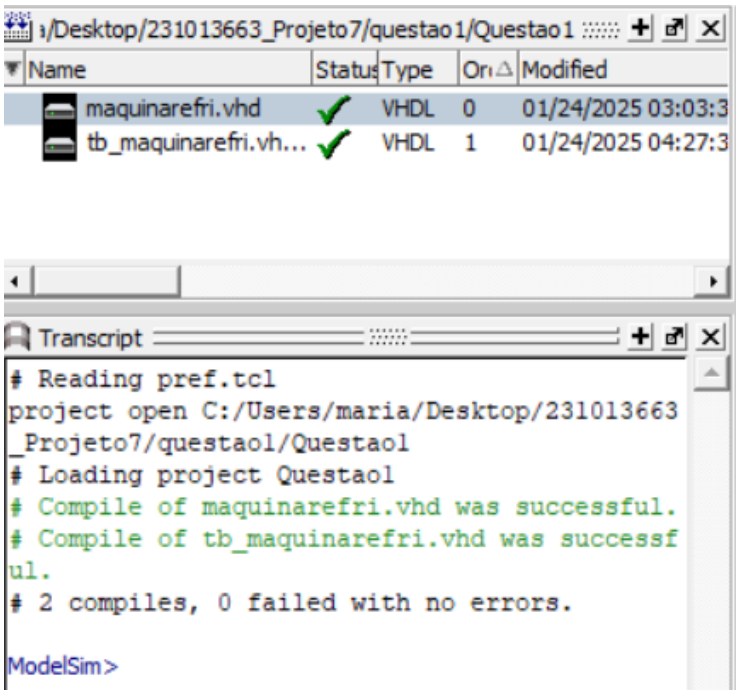

Figura 4. Compilação da questão 1

5. Simulação

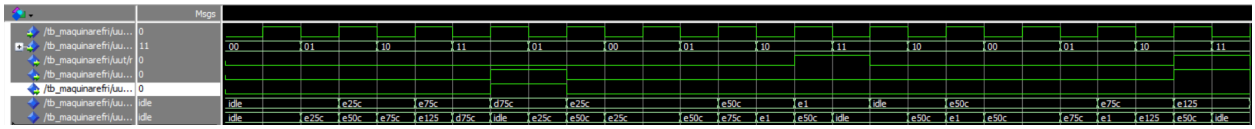

Figura 5. Simulação de onda do banco de testes da questão 1

.

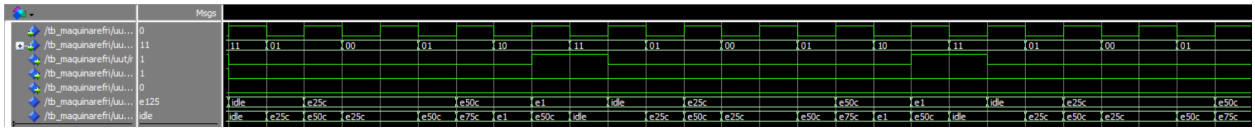

Figura 6. Simulação de onda do banco de testes da questão 1 (Cont)

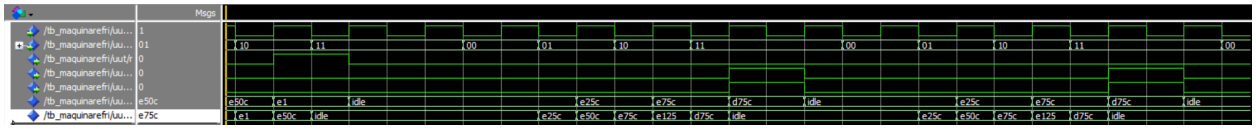

Figura 7. Simulação de onda do banco de testes da questão 1 (Cont)

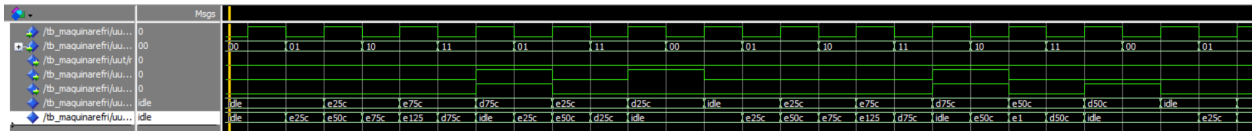

Figura 8. Simulação de onda do banco de testes da questão 1 (Cont)

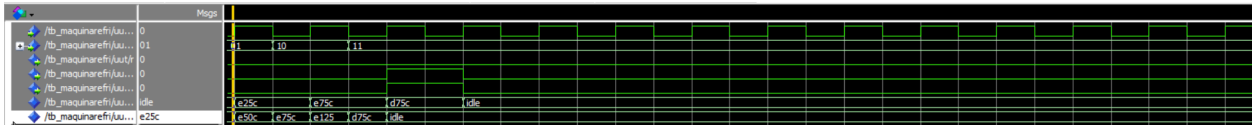

Figura 9. Simulação de onda do banco de testes da questão 1 (Cont)

Para idle

- **Cursor 1 (3.928 ns): clk = 0 , moeda = 00 , r = 0 , c25 = 0 , c50 = 0 , currentstate = idle , nextstate = idle**
- **Cursor 2 (87.51 ns): clk = 1 , moeda = 11 , r = 0 , c25 = 0 , c50 = 0 , currentstate = idle, nextstate = idle**
- **Cursor 3 (13.53 ns): clk = 0 , moeda = 01 , r = 0 , c25 = 0 , c50 = 0 , currentstate = idle, nextstate = e25c**
- **Cursor 4 (92.966 ns): clk = 0 , moeda = 10 , r = 0 , c25 = 0 , c50 = 0 , currentstate = idle, nextstate = e50c**

Para E25C

- **Cursor 5 (17.895 ns): clk = 1 , moeda = 01 , r = 0 , c25 = 0 , c50 = 0 , currentstate = e25c , nextstate = e50c**
- **Cursor 6 (23.132 ns): clk = 0 , moeda = 10 , r = 0 , c25 =0 , c50 = 0 , currentstate = e25c , nextstate = e75c**
- **Cursor 7 (53.466 ns): clk = 0 , moeda = 00 , r = 0 , c25 = 0 , c50 = 0 , currentstate = e25c , nextstate = e25c**
- **Cursor 8 (452.968 ns): clk = 0 , moeda = 11 , r = 0 , c25 = 0 , c50 = 0 , currentstate = e25c , nextstate = d25c**

Para E50C

- **Cursor 9 (67.87 ns): clk = 1 , moeda = 01 , r = 0 , c25 = 0 , c50 = 0 , currentstate = e50c, nextstate = e75c**
- **Cursor 10 (72.889 ns): clk = 0 , moeda = 10 , r = 0 , c25 = 0 , c50 = 0 , currentstate = e50c, nextstate = e1**

- **Cursor 11 (103.441 ns): clk = 0 , moeda = 00 , r = 0 , c25 = 0 , c50 = 0 , currentstate = e50c , nextstate = e50c**
- **Cursor 12 (512.793 ns): clk = 0 , moeda = 11 , r = 0 , c25 = 0 , c50 = 0 , currentstate = e50c , nextstate = d50c**

Para E75C

- **Cursor 13 (27.933 ns): clk = 1 , moeda = 10 , r = 0 , c25 = 0 , c50 = 0 , currentstate = e75c, nextstate = e125**
- **Cursor 14 (32.298 ns): clk = 0 , moeda = 11 , r = 0 , c25 = 0 , c50 = 0 , currentstate = e75c, nextstate = d75c**
- **Cursor 15 (115 ns): clk = 1 , moeda = 01 , r = 0 , c25 = 0 , c50 = 0 , currentstate = e75c , nextstate = e1**

Para E1

- **Cursor 16 (77.69 ns): clk = 1 , moeda = 10 , r = 1 , c25 = 0 , c50 = 0 , currentstate = e1, nextstate = e50c**
- **Cursor 17 (81.4 ns): clk = 0 , moeda = 11 , r = 1 , c25 = 0 , c50 = 0 , currentstate = e1, nextstate = idle**

Para E125

- **Cursor 18 (128.064 ns): clk = 1 , moeda = 10 , r = 1 , c25 = 1 , c50 = 0 , currentstate = e125, nextstate = e50c**
- **Cursor 19 (131.92 ns): clk = 0 , moeda = 11 , r = 1 , c25 = 1 , c50 = 0 , currentstate = e125, nextstate = idle**

Para D25C

- **Cursor 20 (455 ns): clk = 1 , moeda = 11 , r = 0 , c25 = 1 , c50 = 0 , currentstate = d25c, nextstate = idle**
- **Cursor 21 (460.714 ns): clk = 0 , moeda = 00 , r = 0 , c25 = 1 , c50 = 0 , currentstate = d25c, nextstate = idle**

Para D50C

- **Cursor 22 (517.186 ns): clk = 1 , moeda = 11 , r = 0 , c25 = 0 , c50 = 1 , currentstate = d50c, nextstate = idle**
- **Cursor 23 (522.205 ns): clk = 0 , moeda = 00 , r =0 , c25 = 0 , c50 = 1 , currentstate = d50c, nextstate = idle**

Para D75C

- **Cursor 24 (559.304 ns): clk = 1 , moeda = 11 , r = 0 , c25 = 1 , c50 = 1 , currentstate = d75c, nextstate = idle**

- **Cursor 25 (502.128 ns): clk = 0 , moeda = 10 , r = 0 , c25 = 1 , c50 = 1 , currentstate = d75c, nextstate = e50c**
- **Cursor 26 (42.337 ns): clk = 0 , moeda = 01 , r = 0 , c25 = 1 , c50 = 1 , currentstate = d75c, nextstate = e25c**

6. Análise

Neste experimento, foi analisada uma estrutura: a máquina de estados do tipo Moore . Com base na Tabela 1 na simulação de de onda apresentada nas Figuras 5 a 9, é possível compreender o funcionamento esperado do circuito. Dessa forma, pode-se afirmar que os códigos desenvolvidos correspondem ao comportamento esperado, uma vez que os valores obtidos coincidem com aqueles descritos nas tabelas verdade

7. Conclusão

No experimento, foi possível descrever o comportamento da estrutura proposta e entender suas características. As simulações geraram os dados esperados, que foram comparados com a Tabela 1. Não houveram erros ou divergências observados durante a realização do experimento.