Universidade de Brasília
Faculdade de Tecnologia
Laboratório de Sistemas Digitais

Relatório 07

Carla de Araujo Clementino Ribeiro Mat:180030736

Professor:
Guilherme de Sousa Torres

1 Objetivos

Implementar uma máquina de estados do tipo Moore que simula o funcionamento de uma máquina de refrigerantes utilizando VHDL junto ao software ModelSim, além de realizar os testes necessários para comprovar seu funcionamento.

2 Questões Propostas

1. Implementar em VHDL e simular no ModelSim uma máquina de estado síncrona do tipo Moore para controlar uma máquina de refrigerantes que aceita moedas de R$ 0,25 e R$ 0,50. A cada transição do clock, a máquina deve contar o dinheiro inserido e liberar o refrigerante (e o troco) assim que a soma totalizar ou exceder R$ 1,00. A máquina deve aceitar qualquer combinação de moedas de R$ 0,25 e R$ 0,50, independentemente da ordem em que as moedas foram inseridas. A qualquer momento (desde que a contagem ainda não tenha alcançado R$ 1,00) o usuário poderá cancelar a compra e a máquina deve, também na transição do clock, devolver a quantia inserida.

   Considere que a máquina só dispõe de um sabor de refrigerante (ou que a escolha do refrigerante é feita antes da máquina de estados iniciar). Logo, o refrigerante é liberado automaticamente (mas na transição do clock) após a inserção do valor de R$ 1,00 com ou sem troco, não sendo necessário pressionar nenhum botão após a inserção do montante para receber o refrigerante. Isto impede a possibilidade, por exemplo, da inserção do valor de R$ 1,50.

   A entidade VHDL deverá ter como entrada um vetor A de 2 bits que indicará se foi inserida uma moeda de R$ 0,25 (se A = 01), se foi inserida uma moeda de R$ 0,50 (se A = 10), se foi solicitada o cancelamento da compra (se A = 11) ou se não houve nenhuma ação por parte do usuário (se A = 00).

   Deverá ter como entrada também um clock de 1 bit que fará com que a máquina leia as entradas e mude (ou não) o estado e as saídas. A leitura das entradas, assim como a mudança de estado e das saídas, deverá acontecer exclusivamente na borda de subida do clock.

   A entidade VHDL deverá ter três saídas de 1 bit cada que indicarão, respectivamente, se a máquina liberou (ou não) um refrigerante, devolveu (ou não) uma moeda de R$ 0,25 e(ou) devolveu (ou não) uma moeda de R$ 0,50.

   Na estrutura process que descreve a lógica dos registradores (memória da máquina de estados), inclua uma condição de "reset" (associada a uma entrada da entidade, ligada a um botão ou a uma chave do kit de desenvolvimento) que leve, de forma assíncrona (isto é, independentemente do sinal de clock) a máquina de estados de volta ao estado inicial. Isto facilitará o processo de teste da máquina implementada. Um exemplo deimplementação de condição de "reset" é mostrada no documento tutorial fornecido em conjunto a este roteiro.

3 Desenvolvimento Teórico

Considerando o problema descrito acima é possível implementar uma máquina de estados utilizando 9 estados, o quais podem ser descritos por:

1. Estado inicial (sem moedas)
2. Total inserido de 25 centavos
3. Total inserido de 50 centavos
4. Total inserido de 75 centavos
5. Total inserido de 100 centavos (libera o refrigerante)
6. Total inserido de 125 centavos (libera o refrigerante e devolve 25 centavos)
7. Devolver 25 centavos
8. Devolver 50 centavos
9. Devolver 75 centavos

Dessa forma é possível representá-la pelo seguinte diagrama de transição de estados:

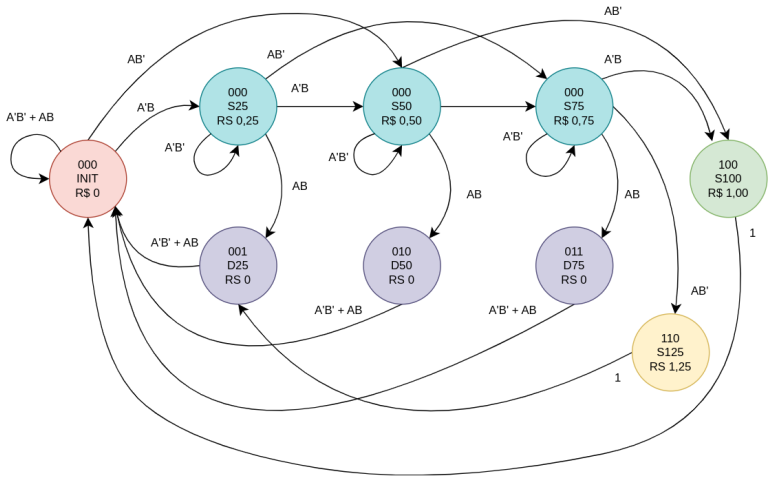

Figura 1: Diagrama de Transição de Estados. Fonte: Autor.

Sendo sua tabela dada por:

	AB				
ESTADO	00	01	10	11	SAÍDA
INIT	INIT	S25	S50	INIT	000
S25	S25	S50	S75	D25	000
S50	S50	S75	S100	D50	000
S75	S75	S100	S125	D75	000
S100	INIT	INIT	INIT	INIT	100
S125	D25	D25	D25	D25	110
D25	INIT	INIT	INIT	INIT	010
D50	INIT	INIT	INIT	INIT	001
D75	INIT	INIT	INIT	INIT	011
	PRÓXIMO ESTADO				

Figura 2: Tabela de Transição de Estados. Fonte: Autor.

4 Simulação

Primeiramente, foi implementada a arquitetura da máquina de estados de acordo com a tabela de transição de estados acima (11), que pode ser vista abaixo:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity maquina is
5      port (
6          A: in std_logic_vector(1 downto 0); -- A[0] = B, A[1] = A
7          clock, rst: in std_logic;
8          refri, devolve_50, devolve_25: out std_logic
9      );
10 end maquina;
11
12 architecture main of maquina is
13     type state_type is (INIT, INSERIU_25, INSERIU_50, INSERIU_75, LIBERA_REFRI, REFRI_25, CANCELA_25,
           CANCELA_50, CANCELA_75);
14     signal current_state, next_state: state_type;
15 begin
16     sync_proc: process(clock, rst)
17     begin
18         if(rst = '1') then
19             current_state <= INIT;
20         elsif rising_edge(clock) then
21             current_state <= next_state;
```

```
22          end if;
23      end process sync_proc;
24
25      combinacao: process(current_state, A)
26      begin
27          case current_state is
28              when INIT =>
29                  refri <= '0';
30                  devolve_50 <= '0';
31                  devolve_25 <= '0';
32                  if (A = "00" or A = "11") then
33                      next_state <= INIT;
34                  elsif (A = "01") then
35                      next_state <= INSERIU_25;
36                  elsif (A = "10") then
37                      next_state <= INSERIU_50;
38                  end if;
39
40              when INSERIU_25 =>
41                  refri <= '0';
42                  devolve_50 <= '0';
43                  devolve_25 <= '0';
44                  if (A = "00") then
45                      next_state <= INSERIU_25;
46                  elsif (A = "01") then
47                      next_state <= INSERIU_50;
48                  elsif (A = "10") then
49                      next_state <= INSERIU_75;
50                  elsif (A = "11") then
51                      next_state <= CANCELA_25;
52                  end if;
53
54              when INSERIU_50 =>
55                  refri <= '0';
56                  devolve_50 <= '0';
57                  devolve_25 <= '0';
58                  if (A = "00") then
59                      next_state <= INSERIU_50;
60                  elsif (A = "01") then
61                      next_state <= INSERIU_75;
62                  elsif (A = "10") then
63                      next_state <= LIBERA_REFRI;
64                  elsif (A = "11") then
65                      next_state <= CANCELA_50;
66                  end if;
67
68              when INSERIU_75 =>
69                  refri <= '0';
70                  devolve_50 <= '0';
71                  devolve_25 <= '0';
72                  if (A = "00") then
73                      next_state <= INSERIU_75;
74                  elsif (A = "01") then
75                      next_state <= LIBERA_REFRI;
76                  elsif (A = "10") then
77                      next_state <= REFRI_25;
78                  elsif (A = "11") then
79                      next_state <= CANCELA_75;
80                  end if;
81
82              when LIBERA_REFRI =>
83                  refri <= '1';
84                  devolve_50 <= '0';
85                  devolve_25 <= '0';
86                  next_state <= INIT;
87
```

```
88              when REFRI_25 =>
89                  refri <= '1';
90                  devolve_50 <= '0';
91                  devolve_25 <= '1';
92                  next_state <= INIT;
93
94              when CANCELA_25 =>
95                  refri <= '0';
96                  devolve_50 <= '0';
97                  devolve_25 <= '1';
98                  next_state <= INIT;
99
100             when CANCELA_50 =>
101                 refri <= '0';
102                 devolve_50 <= '1';
103                 devolve_25 <= '0';
104                 next_state <= INIT;
105
106             when CANCELA_75 =>
107                 refri <= '0';
108                 devolve_50 <= '1';
109                 devolve_25 <= '1';
110                 next_state <= INIT;
111
112             when others =>
113                 refri <= '0';
114                 devolve_50 <= '0';
115                 devolve_25 <= '0';
116                 next_state <= INIT;
117         end case;
118     end process combinacao;
119 end main;
```

Em seguida foram criados 7 arquivos de teste para verificar se a máquina está se comportando corretamente em todos os casos. Para os testes, considere que os sinais são dados por:

- Sinal u_A: Entrada A de 2 bits;
- Sinal u_rst: Sinal de reset;
- Sinal u_clk: Sinal de clock;
- Sinal u_r: Indica a liberação do refrigerante;
- Sinal u_d50: Indica a liberação de uma moeda de 50 centavos;
- Sinal u_25: Indica a liberação de uma moeda de 25 centavos;

4.1 1º Caso

No primeiro teste, verificamos a inserção de uma moeda de 25 centavos e em seguida o cancelamento da operação. A operação deve retornar uma moeda de 25 centavos e em seguida voltar ao estado inicial.

O teste é dado pelo seguinte código:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity test_bench is
5  end entity;
6
7  architecture main of test_bench is
8      component maquina is
9          port (
10             A: in std_logic_vector (1 downto 0); -- A[0] = B A[1] = B
11             clock, rst: in std_logic;
12             refri, devolve_50, devolve_25 : out std_logic
13         );
14     end component;
```

```
15
16      signal u_A : std_logic_vector(1 downto 0) := "00";
17      signal u_rst, u_clk, r, d_50, d_25: std_logic := '0';
18
19  begin
20      tb_shift_register: maquina
21          port map(
22              A => u_A,
23              rst => u_rst,
24              clock => u_clk,
25              refri => r,
26              devolve_50 => d_50,
27              devolve_25 => d_25
28          );
29
30      process
31      begin
32          while true loop
33              u_clk <= '0';
34              wait for 1 ns;
35              u_clk <= '1';
36              wait for 1 ns;
37          end loop;
38      end process;
39
40      process
41      begin
42          u_A <= "00" after 0 ns,
43                 "01" after 2 ns,
44                 "11" after 4 ns;
45          wait;
46      end process;
47
48  end architecture;
```

O resultado pode ser visto abaixo:

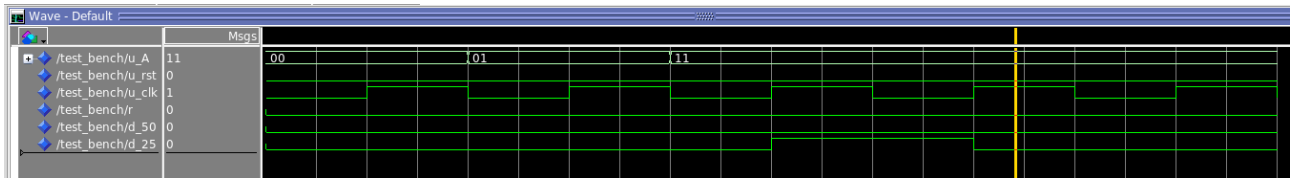

Figura 3: Retorna 25 centavos. Fonte: Autor.

4.2 2º Caso

No segundo teste, verificamos a inserção de uma moeda de 50 centavos e em seguida o cancelamento da operação. A operação deve retornar uma moeda de 50 centavos e em seguida voltar ao estado inicial.

O teste é dado pelo seguinte código:

```
1
2   -- Cancela apos inserir 50 centavos
3
4   library IEEE;
5   use IEEE.STD_LOGIC_1164.ALL;
6
7   entity test_bench is
8   end entity;
9
10  architecture main of test_bench is
11      component maquina is
12          port (
13              A: in std_logic_vector (1 downto 0); -- A[0] = B A[1] = B
14              clock, rst: in std_logic;
```

```
15              refri, devolve_50, devolve_25 : out std_logic
16          );
17      end component;
18
19      signal u_A : std_logic_vector(1 downto 0) := "00";
20      signal u_rst, u_clk, r, d_50, d_25: std_logic := '0';
21
22  begin
23      tb_shift_register: maquina
24          port map(
25              A => u_A,
26              rst => u_rst,
27              clock => u_clk,
28              refri => r,
29              devolve_50 => d_50,
30              devolve_25 => d_25
31          );
32
33
34      process
35      begin
36          while True loop
37              u_clk <= '0';
38              wait for 1 ns;
39              u_clk <= '1';
40              wait for 1 ns;
41          end loop;
42      end process;
43
44      process
45      begin
46          u_A <= "00" after 0 ns,
47                 "10" after 2 ns,
48                 "11" after 4 ns;
49          wait;
50      end process;
51
52  end architecture;
```

O resultado pode ser visto abaixo:

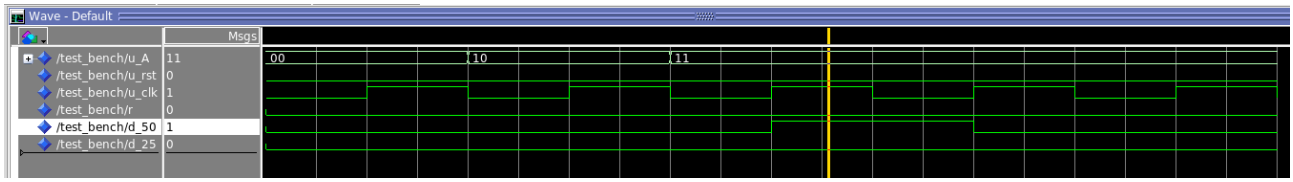

Figura 4: Retorna 50 centavos. Fonte: Autor.

4.3 3º Caso

No terceiro teste, verificamos a inserção de uma moeda de 50 centavos e uma de 25 centavos e em seguida o cancelamento da operação. A operação deve retornar uma moeda de 50 centavos e uma de 25 centavos e, em seguida, voltar ao estado inicial.

O teste é dado pelo seguinte código:

```
1
2   -- Cancela apos inserir 75 centavos
3
4   library IEEE;
5   use IEEE.STD_LOGIC_1164.ALL;
6
7   entity test_bench is
8   end entity;
9
```

```
10 architecture main of test_bench is
11     component maquina is
12         port (
13             A: in std_logic_vector (1 downto 0); -- A[0] = B A[1] = B
14             clock, rst: in std_logic;
15             refri, devolve_50, devolve_25 : out std_logic
16         );
17     end component;
18 
19     signal u_A : std_logic_vector(1 downto 0) := "00";
20     signal u_rst, u_clk, r, d_50, d_25: std_logic := '0';
21 
22 begin
23     tb_shift_register: maquina
24         port map(
25             A => u_A,
26             rst => u_rst,
27             clock => u_clk,
28             refri => r,
29             devolve_50 => d_50,
30             devolve_25 => d_25
31         );
32 
33 
34     process
35     begin
36         while True loop
37             u_clk <= '0';
38             wait for 1 ns;
39             u_clk <= '1';
40             wait for 1 ns;
41         end loop;
42     end process;
43 
44     process
45     begin
46         u_A <= "00" after 0 ns,
47               "10" after 2 ns,
48               "01" after 4 ns,
49               "11" after 6 ns;
50         wait;
51     end process;
52 
53 end architecture;
```

O resultado pode ser visto abaixo:

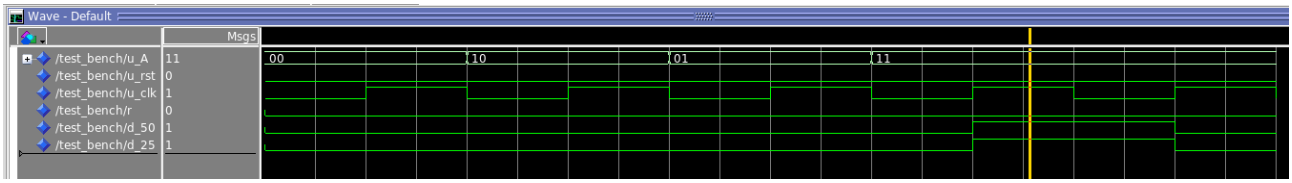

Figura 5: Retorna 75 centavos. Fonte: Autor.

4.4 4º Caso

No quarto teste, verificamos a inserção de duas moeda de 50 centavos. A operação deve liberar o refrigerante e voltar ao estado inicial.

O teste é dado pelo seguinte código:

```
1     -- Recebe o refri apos inserir 2 moedas de 50 centavos
2 
3 library IEEE;
4 use IEEE.STD_LOGIC_1164.ALL;
```

```
5
6  entity test_bench is
7  end entity;
8
9  architecture main of test_bench is
10     component maquina is
11         port (
12             A: in std_logic_vector (1 downto 0); -- A[0] = B A[1] = B
13             clock, rst: in std_logic;
14             refri, devolve_50, devolve_25 : out std_logic
15         );
16     end component;
17
18     signal u_A : std_logic_vector(1 downto 0) := "00";
19     signal u_rst, u_clk, r, d_50, d_25: std_logic := '0';
20
21 begin
22     tb_shift_register: maquina
23         port map(
24             A => u_A,
25             rst => u_rst,
26             clock => u_clk,
27             refri => r,
28             devolve_50 => d_50,
29             devolve_25 => d_25
30         );
31
32
33     process
34     begin
35         while True loop
36             u_clk <= '0';
37             wait for 1 ns;
38             u_clk <= '1';
39             wait for 1 ns;
40         end loop;
41     end process;
42
43     process
44     begin
45         u_A <= "00" after 0 ns,
46                "10" after 2 ns,
47                "10" after 4 ns;
48         wait;
49     end process;
50
51 end architecture;
```

O resultado pode ser visto abaixo:

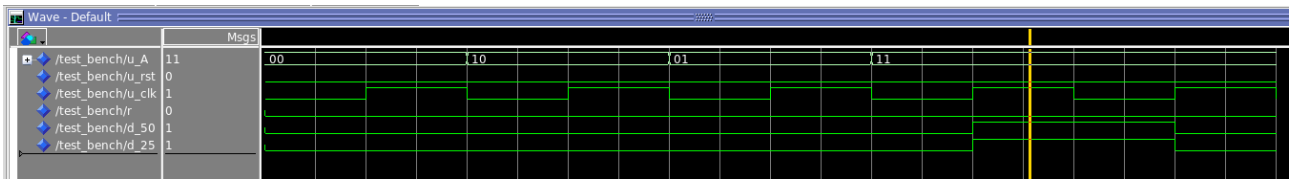

Figura 6: Libera o refrigerante - 1º caso. Fonte: Autor.

4.5 5º Caso

No quinto teste, verificamos a inserção de uma moeda de 50 centavos e duas de 25 centavos. A operação deve liberar o refrigerante e voltar ao estado inicial.

O teste é dado pelo seguinte código:

```
1  -- Recebe o refri depois de inserir 2 moedas de 25 centavos e uma de 50 centavos
```

```
2
3  library IEEE;
4  use IEEE.STD_LOGIC_1164.ALL;
5
6  entity test_bench is
7  end entity;
8
9  architecture main of test_bench is
10     component maquina is
11         port (
12             A: in std_logic_vector (1 downto 0); -- A[0] = B A[1] = B
13             clock, rst: in std_logic;
14             refri, devolve_50, devolve_25 : out std_logic
15         );
16     end component;
17
18     signal u_A : std_logic_vector(1 downto 0) := "00";
19     signal u_rst, u_clk, r, d_50, d_25: std_logic := '0';
20
21 begin
22     tb_shift_register: maquina
23         port map(
24             A => u_A,
25             rst => u_rst,
26             clock => u_clk,
27             refri => r,
28             devolve_50 => d_50,
29             devolve_25 => d_25
30         );
31
32
33     process
34     begin
35         while True loop
36             u_clk <= '0';
37             wait for 1 ns;
38             u_clk <= '1';
39             wait for 1 ns;
40         end loop;
41     end process;
42
43     process
44     begin
45         u_A <= "00" after 0 ns,
46                "10" after 2 ns,
47                "01" after 4 ns,
48                "01" after 6 ns;
49         wait;
50     end process;
51
52 end architecture;
```

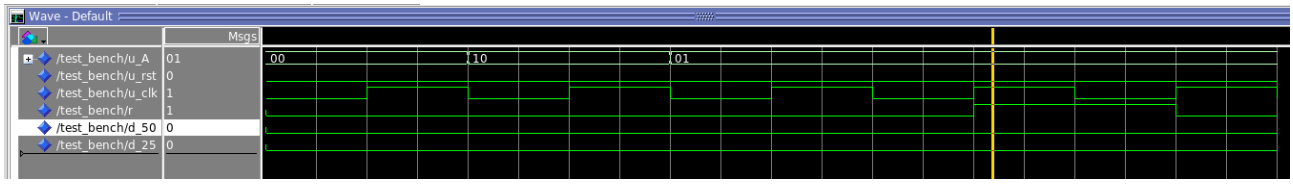

Figura 7: Libera o refrigerante - 2º caso. Fonte: Autor.

4.6 6º Caso

No quinto teste, verificamos a inserção de quatro moeda de 25 centavos. A operação deve liberar o refrigerante e voltar ao estado inicial.

O teste é dado pelo seguinte código:

```
1
2  -- Recebe o refri depois de inserir 4 moedas de 25 centavos
3
4  library IEEE;
5  use IEEE.STD_LOGIC_1164.ALL;
6
7  entity test_bench is
8  end entity;
9
10 architecture main of test_bench is
11     component maquina is
12         port (
13             A: in std_logic_vector (1 downto 0); -- A[0] = B A[1] = B
14             clock, rst: in std_logic;
15             refri, devolve_50, devolve_25 : out std_logic
16         );
17     end component;
18
19     signal u_A : std_logic_vector(1 downto 0) := "00";
20     signal u_rst, u_clk, r, d_50, d_25: std_logic := '0';
21
22 begin
23     tb_shift_register: maquina
24         port map(
25             A => u_A,
26             rst => u_rst,
27             clock => u_clk,
28             refri => r,
29             devolve_50 => d_50,
30             devolve_25 => d_25
31         );
32
33
34     process
35     begin
36         while True loop
37             u_clk <= '0';
38             wait for 1 ns;
39             u_clk <= '1';
40             wait for 1 ns;
41         end loop;
42     end process;
43
44     process
45     begin
46         u_A <= "00" after 0 ns,
47                "01" after 2 ns,
48                "01" after 4 ns,
49                "01" after 6 ns,
50                "01" after 8 ns;
51         wait;
52     end process;
53
54 end architecture;
```

O resultado pode ser visto abaixo:

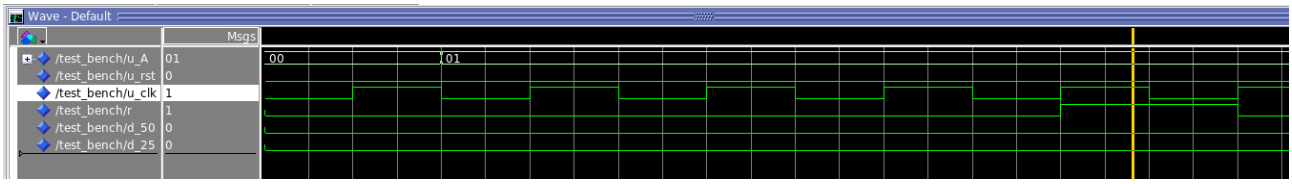

Figura 8: Libera o refrigerante - 3º caso. Fonte: Autor.

4.7 6º Caso

No sexto teste, verificamos a inserção de 1 moeda de 50 centavos seguida de uma de 25 e outra de 50, respectivamente. A operação deve liberar o refrigerante, retornar 25 centavos e voltar ao estado inicial.

O teste é dado pelo seguinte código:

```
-- Recebe o refri depois de inserir 2 moedas de 50 centavos e 1 de 25 centavos

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity test_bench is
end entity;

architecture main of test_bench is
    component maquina is
        port (
            A: in std_logic_vector (1 downto 0); -- A[0] = B A[1] = B
            clock, rst: in std_logic;
            refri, devolve_50, devolve_25 : out std_logic
        );
    end component;

    signal u_A : std_logic_vector(1 downto 0) := "00";
    signal u_rst, u_clk, r, d_50, d_25: std_logic := '0';

begin
    tb_shift_register: maquina
        port map(
            A => u_A,
            rst => u_rst,
            clock => u_clk,
            refri => r,
            devolve_50 => d_50,
            devolve_25 => d_25
        );


    process
    begin
        while True loop
            u_clk <= '0';
            wait for 1 ns;
            u_clk <= '1';
            wait for 1 ns;
        end loop;
    end process;

    process
    begin
        u_A <= "00" after 0 ns,
               "10" after 2 ns,
               "01" after 4 ns,
               "10" after 6 ns;
        wait;
    end process;

end architecture;
```

O resultado pode ser visto abaixo:

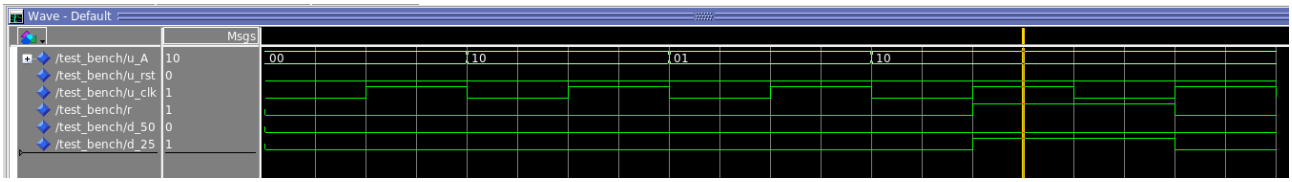

Figura 9: Libera o refrigerante - 4º caso. Fonte: Autor.

4.8 7º Caso

No sétimo teste, verificamos o funcionamento do sinal de reset.
O teste é dado pelo seguinte código:

```
    -- Testa o sinal de reset
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity test_bench is
end entity;

architecture main of test_bench is
    component maquina is
        port (
            A: in std_logic_vector (1 downto 0); -- A[0] = B A[1] = B
            clock, rst: in std_logic;
            refri, devolve_50, devolve_25 : out std_logic
        );
    end component;

    signal u_A : std_logic_vector(1 downto 0) := "00";
    signal u_rst, u_clk, r, d_50, d_25: std_logic := '0';

begin
    tb_shift_register: maquina
        port map(
            A => u_A,
            rst => u_rst,
            clock => u_clk,
            refri => r,
            devolve_50 => d_50,
            devolve_25 => d_25
        );

    process
    begin
        while True loop
            u_clk <= '0';
            wait for 1 ns;
            u_clk <= '1';
            wait for 1 ns;
        end loop;
    end process;

    process
    begin
        u_rst <= '1' after 6 ns;
        u_A <= "01" after 2 ns,
               "10" after 4 ns;
        wait;
    end process;

end architecture;
```

O resultado pode ser visto abaixo:

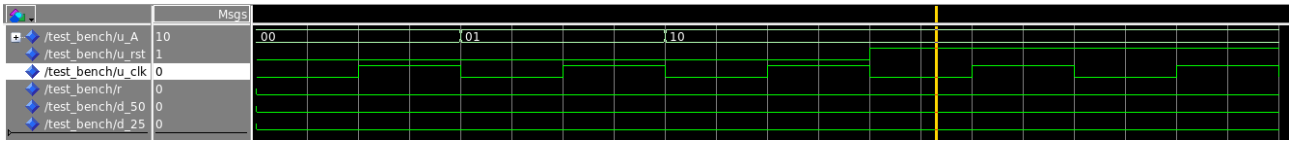

Figura 10: Sinal de reset. Fonte: Autor.

5 Compilação dos Arquivos

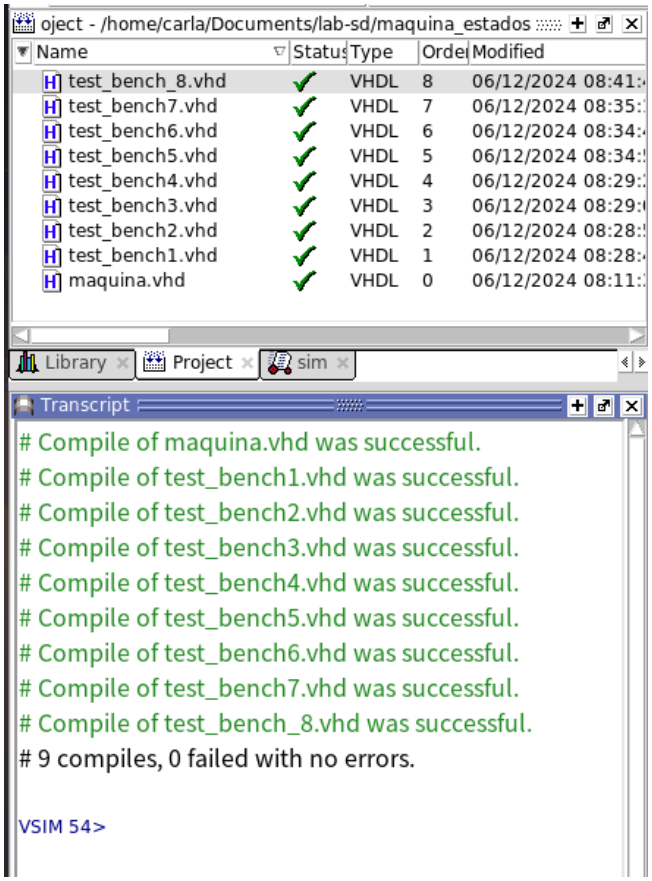

Figura 11: Compilação dos Arquivos. Fonte: Autor.

6 Conclusão

Considerando a implementação feita, é possível afirmar que o projeto prioriza a qualidade sobre o custo, considerando que foi escolhido tratar todos os cenários possíveis. Essa decisão foi escolhida tendo em vista uma maior segurança e facilidade na manutenção do projeto.