PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS - PUC CAMPINAS
CEATEC - CENTRO DE CIÊNCIAS EXATAS, AMBIENTAIS E DE TECNOLOGIA

Projeto 2 – Arquitetura de Computadores

CPU - Simples

Fabio Luis Dumont - RA 17049461

Marcos Lelis - RA 16248387

Rafael Alves de Oliveira Perroni- RA 18009340

Victor Luiz Fraga Soldera- RA 18045674

Campinas-SP 2019

Índice

1.Descrição da Topologia da CPU

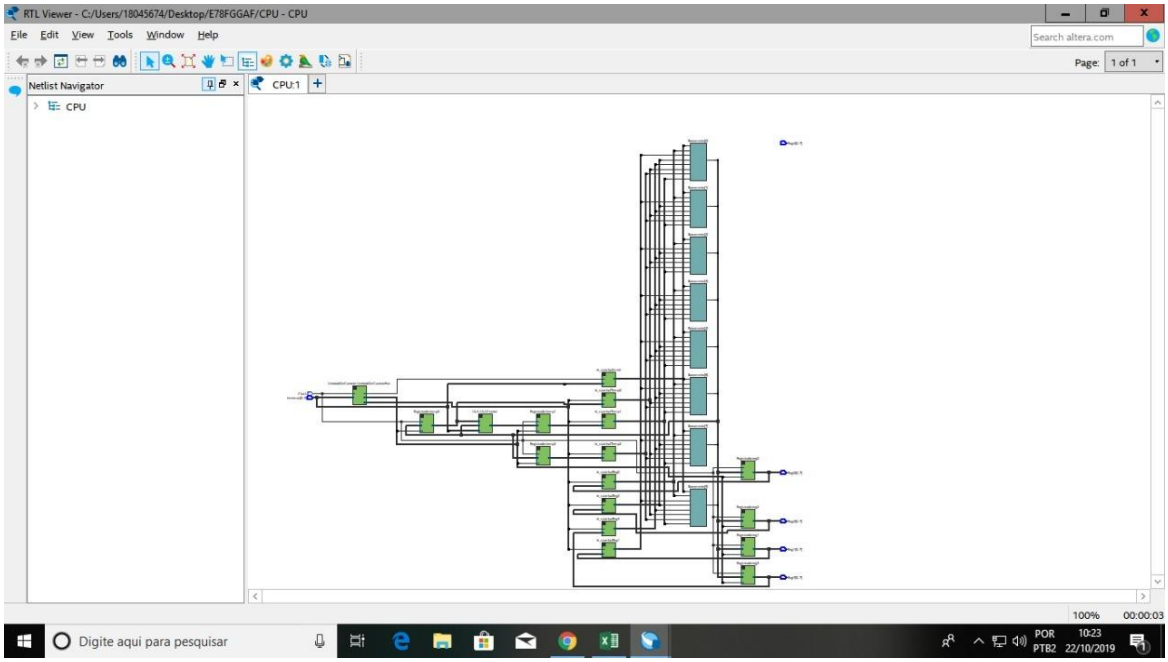
Nesse segundo projeto de Arquitetura de Computadores foi proposto o desenvolvimento de uma CPU simples, capaz de executar as seguintes operações:

Instrução	Significado	Descrição
MOV Ri, Rj	Ri <- Rj	Move
MOVI Ri, Imed	Ri <- Imed	Move Immediate
XCHG Ri, Rj	Ri <- Rj e Rj <- Ri	Exchange
ADD Ri, Rj, Rk	Ri <- Rj + Ri	Add
ADDI Ri, Rj, Imed	Ri <- Rj + Imed	Add Immediate
SUB Ri, Rj, Rk	Ri <- Rj - Ri	Subtract
SUBI Ri, Rj, Imed	Ri <- Rj - Imed	Subtract Immediate
AND Ri, Rj, Rk	Ri <- Rj & Ri	And
ANDI Ri, Rj, Imed	Ri <- Rj & Imed	And Immediate
OR Ri, Rj, Rk	Ri <- Rj | Ri	Or
ORI Ri, Imed	Ri <- Ri | Imed	Or Immediate

Para isso o projeto foi baseado no livro “Fundamentals of Digital Logic with VHDL Design”, mais especificamente no capítulo 7 do livro. Com o livro foi possível o entendimento e o desenvolvimento da CPU.

A CPU espera por uma entrada de instrução que é computada pela Unidade de Controle, para que seja realizada a operação requisitada, além disso usamos o Quartus como programa para desenvolver o VHDL e realizar testes de Waveform para a comprovação dos resultados aguardados.

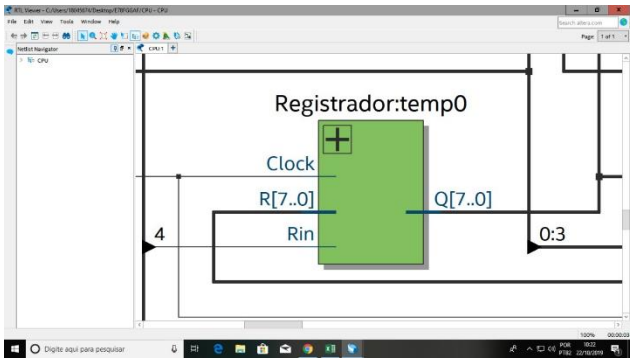
Essa simulação feita pelo RTL Viewer (opção presente no software Quartus), representa a CPU projetada:



2. Especificações

Registradores:

Projetou-se um CPU com 4 registradores de 8 bits cada, sendo 3 deles para realização das operações na ULA (Unidade Lógica Aritmética) e um deles para a realização do XCHG, para que não haja conflito de informações entre eles no decorrer da execução, sendo todos os registradores endereçados.



Formato das Instruções

As instruções são de dois tipos, tipo R e tipo I, ou seja, são referenciadas de diferentes formas de modo que, o tipo R é de 4 bits para OPcode e 3 registradores com 2 bits cada, totalizando 10 bits por instrução. Já as de tipo I

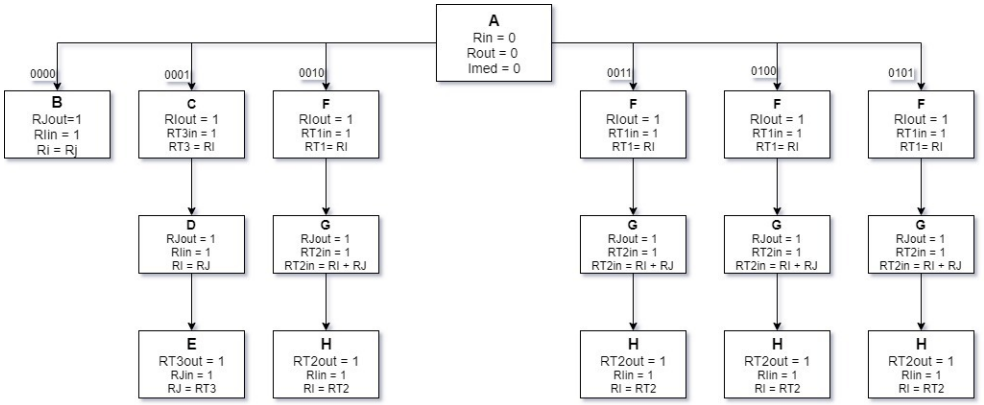
são 4 bits para o OPcode, 2bits para o registrador e 4bits para o imediato, totalizando os mesmos 10 bits por instrução.

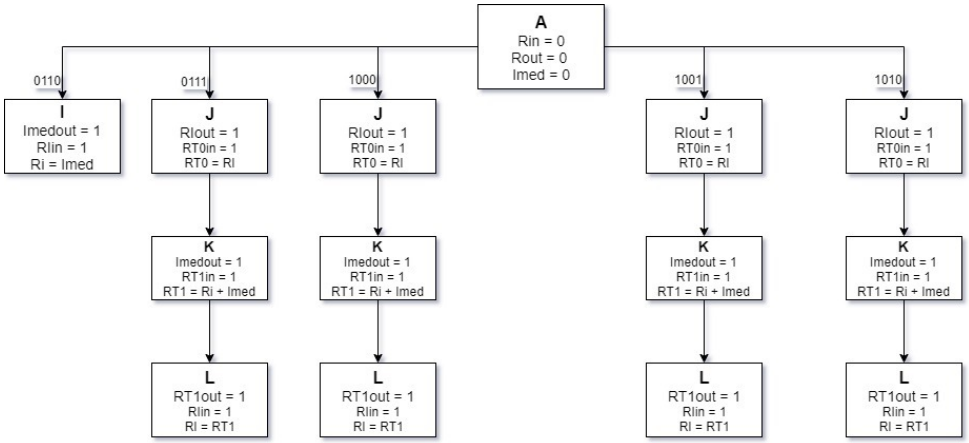
Instrução	OPcode	Tipos
MOV Ri, Rj	0000	Tipo R/I
MOVI Ri, Imed	0110	Tipo I
XCHG Ri, Rj	0001	Tipo R/I
ADD Ri, Rj, Rk	0010	Tipo R
ADDI Ri, Rj, Imed	0111	Tipo I
SUB Ri, Rj, Rk	0011	Tipo R
SUBI Ri, Rj, Imed	1000	Tipo I
AND Ri, Rj, Rk	0100	Tipo R
ANDI Ri, Rj, Imed	1001	Tipo I
OR Ri, Rj, Rk	0101	Tipo R
ORI Ri, Imed	1010	Tipo I

A tabela acima mostra todas as instruções, OPcodes e tipos de instrução, respectivamente.

Unidade de Controle

O diagrama de estados colocado abaixo demonstra a unidade de controle.



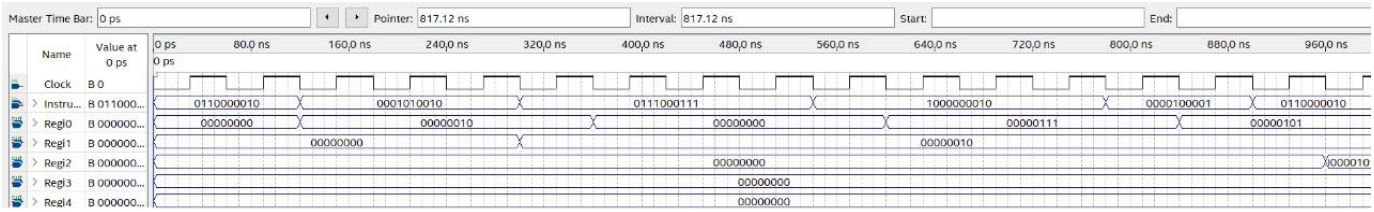


Demonstrando todos os sinais usados em cada operação na requisição de cada componente da CPU, no controle do conjunto de componentes implementados com sua devida função.

Tendo os componentes sensíveis ao clock para que funcionem de forma adequada e sem conflitos entre os sinais nos registradores e barramento, ou seja, quando há o sinal de Reg, significa um fluxo In no registrador em questão, já nos Imed quando 1 controla os tri state buffer para a entrada de imediatos na operação, e ALU com seus diferentes conjuntos de bits para cada operação.

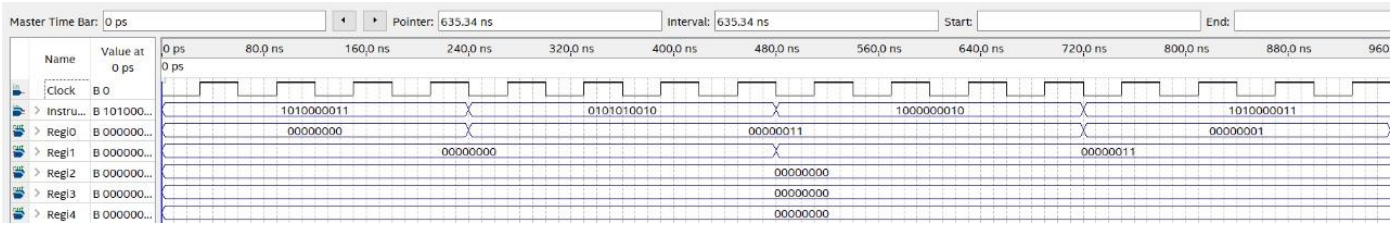
3. Resultados

Abaixo segue prints dos resultados obtidos em simulações waveform.

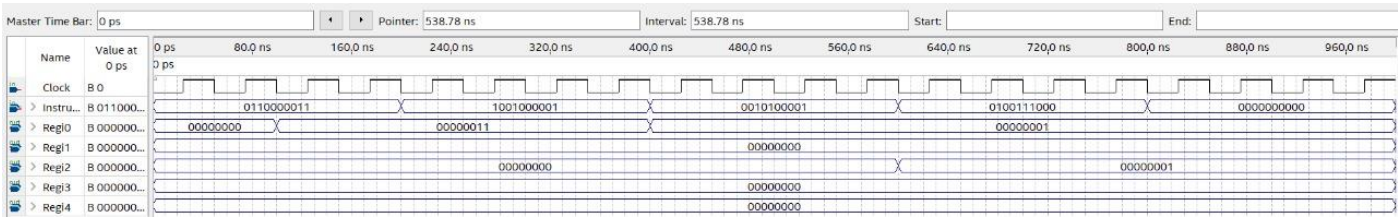


A imagem acima mostra o teste de 5 operações: MOVI, XCHG, ADDI, SUB, MOV, em respectiva ordem na imagem, com todos os testes tendo os resultados esperados.

Como pode ser visto na imagem.



Na imagem acima o teste de ORI, OR, SUB e SUBI, respectivamente. Os testes para essas operações foram um sucesso, podendo ser verificado na imagem.



Na imagem acima MOVI, ANDI, ADD e AND, respectivamente.

Todos os testes de todas as operações tiveram êxito de forma que a CPU funciona de forma correta em todas as operações propostas. Na simulação do waveform foi-se passa a instrução de 10 bits e analisada seu retorno no registrador target em todas as operações, como mostrado nas imagens a cima.

Resultados e Discussão

Os resultados obtidos foram muito satisfatórios, sem nenhum problema aparente no projeto final. Mesmo com algumas dificuldades em projetar a CPU e de utilizar-se o waveform de maneira correta, foi-se possível uma análise dos resultados de todas as instruções pedidas para serem implementadas na CPU.

Bibliografia

- Brown S., Vranesic S. “Fundamentals of Digital Logic with VHDL Design”, Capítulo 7 – Seção 7.14: Design Example;

Anexos

CPU.vhd

```
1   library ieee;
2   USE ieee.std_logic_1164.all;
3   USE ieee.std_logic_signed.all ;
4   use ieee.numeric_std.all;
5
6
7   ENTITY CPU IS
8   PORT (
9       Clock : IN STD_LOGIC ;
10      Instrucao : IN STD_LOGIC_VECTOR(0 to 9);
11      Regi0, Regi1, Regi2, Regi3,Regi4     : OUT STD_LOGIC_VECTOR(0 TO 7)
12
13
14      );
15  END CPU;
16
17  -- vetor in e out R0 R1 R2 R3 T0 T1 T2
18  ARCHITECTURE CPUTest OF CPU IS
19
20   -- componentes da UC
21  COMPONENT UnidadeDeControle
22      Port (
23      Clock : IN STD_LOGIC;
24      Instrucao : IN STD_LOGIC_VECTOR(0 to 9);
25      ImedOut : OUT STD_LOGIC;
26      ROut : OUT STD_LOGIC_VECTOR (0 to 7);
27      RIn : OUT STD_LOGIC_VECTOR (0 to 7));
28  END COMPONENT ;
29
30  -- componentes do registrador
31  COMPONENT reg
32      Port (  R: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
33                  Rin, Clock: IN STD_LOGIC;
34                  Q: OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
35  END COMPONENT ;
36
37  -- componentes do buffer
38  COMPONENT tri_state_buffer
39      Port ( Entradas  : in  STD_LOGIC_VECTOR (7 downto 0);
40             ENABLE  : in  STD_LOGIC;
41             Saidas : out STD_LOGIC_VECTOR (7 downto 0));
42  END COMPONENT ;
43
44
45  -- componentes do ula
46  COMPONENT ULA
47      Port (
48     A:       IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
49      B:         IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
50      OP:    IN  STD_LOGIC_VECTOR (3 DOWNTO 0);
51      SAIDA:  OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
```

```
-- componentes do ula
COMPONENT ULA
    Port (
   A:       IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
    B:         IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
    OP:    IN  STD_LOGIC_VECTOR (3 DOWNTO 0);
    SAIDA:  OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
);
END COMPONENT ;

        SIGNAL ImedOut : STD_LOGIC;
        SIGNAL Rin, Rout, Q : STD_LOGIC_VECTOR(0 TO 7) ;
        SIGNAL R2,R3,T0,T1,T2,Barramento,imedAux,R1, AuxT    : STD_LOGIC_VECTOR(0 TO 7) := "00000000";
        SIGNAL R0    : STD_LOGIC_VECTOR(0 TO 7) := "00000001";
BEGIN



imedAux(0) <= Instrucao(6);
imedAux(1) <= Instrucao(6);
imedAux(2) <= Instrucao(6);
imedAux(3) <= Instrucao(6);
imedAux(4) <= Instrucao(6);
imedAux(5) <= Instrucao(7);
imedAux(6) <= Instrucao(8);
imedAux(7) <= Instrucao(9);



-- unidade de controle
UnidadeDeControlePort : UnidadeDeControle PORT MAP (Clock, Instrucao, ImedOut, Rout, Rin);


-- registradores
reg0: reg PORT MAP (Barramento, Rin(0),Clock, R0);
reg1: reg PORT MAP (Barramento, Rin(1),Clock, R1);
reg2: reg PORT MAP (Barramento, Rin(2),Clock, R2);
reg3: reg PORT MAP (Barramento, Rin(3),Clock, R3);
temp0: reg PORT MAP (Barramento, Rin(4),Clock, T0);
temp2: reg PORT MAP (Barramento, Rin(6),Clock, T2);


-- buffer
bufImed: tri_state_buffer PORT MAP  (imedAux, ImedOut, Barramento);

bufReg0: tri_state_buffer PORT MAP  (R0, Rout(0), Barramento);
bufReg1:    tri_state_buffer PORT MAP  (R1, Rout(1), Barramento);
bufReg2:    tri_state_buffer PORT MAP  (R2, Rout(2), Barramento);
bufReg3:    tri_state_buffer PORT MAP  (R3, Rout(3), Barramento);
bufTemp0:   tri_state_buffer PORT MAP  (T0, Rout(4), Barramento);
bufTemp2:   tri_state_buffer PORT MAP  (T2, Rout(6), Barramento);

Regi0 <= R0;
Regi1 <= R1;
Regi2 <= R2;
Regi3 <= R3;

ULAControl: ULA PORT MAP (T0, Barramento, Instrucao(0 to 3), AuxT);
temp1: reg PORT MAP (AuxT, Rin(5),Clock, T1);
bufTemp1:   tri_state_buffer PORT MAP  (T1, Rout(5), Barramento);

END CPUTest;
```

reg.vhd

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4
5  ENTITY reg IS
6
7  PORT(
8  R: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
9  Rin, Clock: IN STD_LOGIC;
10 Q: OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
11 END reg;
12
13 ARCHITECTURE Behavior OF reg IS
14 BEGIN
15 PROCESS (Clock, Rin)
16 BEGIN
17 IF Clock'EVENT AND Clock = '0' AND Rin = '1' THEN
18 Q <= R;
19 END IF;
20 END PROCESS;
21 END Behavior;
```

tri_state_buffer.vhd

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4
5  entity tri_state_buffer is
6      Port ( Entradas  : in  STD_LOGIC_VECTOR (7 downto 0);
7             ENABLE   : in  STD_LOGIC;
8             Saidas : out STD_LOGIC_VECTOR (7 downto 0));
9  end tri_state_buffer;
10
11 architecture Behavioral of tri_state_buffer is
12
13 begin
14
15     Saidas <= (OTHERS => 'Z') WHEN (ENABLE = '0') else Entra
16
17 end Behavioral;
```

ULA.vhd

```
1  LIBRARY ieee ;
2  USE ieee.std_logic_1164.all ;
3  USE ieee.std_logic_signed.all ;
4  use ieee.numeric_std.all;
5
6  ENTITY ULA IS
7  PORT (
8
9     A:       IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
10     B:       IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
11     OP:    IN  STD_LOGIC_VECTOR (3 DOWNTO 0);
12     SAIDA:  OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
13 );
14 END ULA;
15
16 ARCHITECTURE Estrutura OF ULA IS
17
18 BEGIN
19
20 WITH OP SELECT
21     SAIDA <= A + B WHEN "0010",
22                 A + B WHEN "0111",
23                 A - B WHEN "0011",
24                 A - B WHEN "1000",
25                 A and B WHEN "0100",
26                 A and B WHEN "1001",
27                 A OR B WHEN "0101",
28                 A OR B WHEN "1010",
29                 "ZZZZZZZZ" WHEN OTHERS;
30
31 END ESTRUTURA;
32
33
```

Unidade de Controle.vhd

```
1   library ieee;
2   USE ieee.std_logic_1164.all;
3
4
5   ENTITY UnidadeDeControle IS
6   PORT (
7       Clock : IN STD_LOGIC;
8       Instrucao : IN STD_LOGIC_VECTOR(0 to 9);
9       ImedOut : OUT STD_LOGIC;
10      ROut : OUT STD_LOGIC_VECTOR (0 to 7);
11      RIn : OUT STD_LOGIC_VECTOR (0 to 7));
12  END UnidadeDeControle;
13
14  ARCHITECTURE UnidadeDeControleTeste OF UnidadeDeControle IS
15  TYPE State_type IS (A,B,C,D,E,F,G,H,I,J,K,L);
16  SIGNAL estado, prox, prox2 : State_type := A;
17
18  BEGIN
19  PROCESS (Clock )
20  VARIABLE indice,indice2: integer;
21
22  BEGIN
23      IF (Clock'EVENT AND Clock = '1') THEN
24      ROut <= "00000000";
25      RIn <= "00000000";
26      ImedOut <= '0';
27
28      -- indice 1 i indice 2 j
29          CASE estado IS
30          -- se estiver em A
31              WHEN A =>
32                  IF (Instrucao (0 to 3) = "0000") THEN prox <= B;
33                  ELSIF (Instrucao (0 to 3) = "0001") THEN prox <= C;
34                  ELSIF (Instrucao (0 to 3) = "0010")THEN prox <= F;
35                  ELSIF (Instrucao (0 to 3) = "0011") THEN prox <= F;
36                  ELSIF (Instrucao (0 to 3) = "0100") THEN prox <= F;
37                  ELSIF (Instrucao (0 to 3) = "0101") THEN prox <= F;
38                  ELSIF (Instrucao (0 to 3) = "0110") THEN prox <= I;
39                  ELSIF (Instrucao (0 to 3) = "0111") THEN prox <= J;
40                  ELSIF (Instrucao (0 to 3) = "1000") THEN prox <= J;
41                  ELSIF (Instrucao (0 to 3) = "1001") THEN prox <= J;
42                  ELSIF (Instrucao (0 to 3) = "1010") THEN prox <= J;
43                  ELSE prox <= A;
44                  END IF;
45
46                  IF (Instrucao (4 to 5) = "00") THEN indice := 0;
47                  ELSIF (Instrucao (4 to 5) = "01") THEN indice := 1;
48                  ELSIF (Instrucao (4 to 5) = "10") THEN indice := 2;
49                  ELSIF (Instrucao (4 to 5) = "11") THEN indice := 3;
50                  END IF;
51
```

```
52                  IF (Instrucao (6 to 7) = "00") THEN indice2 := 0;
53                  ELSIF (Instrucao (6 to 7) = "01") THEN indice2 := 1;
54                  ELSIF (Instrucao (6 to 7) = "10") THEN indice2 := 2;
55                  ELSIF (Instrucao (6 to 7) = "11") THEN indice2 := 3;
56                  END IF;
57
58                  Rout <= "00000000";
59                  Rin <= "00000000";
60                  ImedOut <= '0';
61
62
63
64                  WHEN B =>
65                      Rout(indice2) <= '1';
66                      Rin(indice) <= '1';
67                      prox <= A;
68
69                  WHEN C =>
70                      Rout(indice) <= '1';
71                      Rin(6) <= '1';
72                      prox <= D;
73
74                  WHEN D =>
75                      Rout(indice2) <= '1';
76                      Rin(indice) <= '1';
77                      prox <= E;
78
79                  WHEN E =>
80                      Rout(6) <= '1';
81                      Rin(indice2) <= '1';
82                      prox <= A;
83
84                  WHEN F =>
85                      Rout(indice) <= '1';
86                      Rin(4) <= '1';
87                      prox <= G;
88
89                  WHEN G =>
90                      Rout(indice2) <= '1';
91                      Rin(5) <= '1';
92                      prox <= H;
93
94                  WHEN H =>
95                      Rout(5) <= '1';
96                      Rin(indice) <= '1';
97                      prox <= A;
98
99                  WHEN I =>
100                     imedOut <= '1';
101                     Rin(indice) <= '1';
102                     prox <= A;
```

```
83
84                      WHEN F =>
85                          Rout(indice) <= '1';
86                          Rin(4) <= '1';
87                          prox <= G;
88
89                      WHEN G =>
90                          Rout(indice2) <= '1';
91                          Rin(5) <= '1';
92                          prox <= H;
93
94                      WHEN H =>
95                          Rout(5) <= '1';
96                          Rin(indice) <= '1';
97                          prox <= A;
98
99                      WHEN I =>
100                         imedOut <= '1';
101                         Rin(indice) <= '1';
102                         prox <= A;
103
104                     WHEN J =>
105                         Rout(indice) <= '1';
106                         Rin(4) <= '1';
107                         prox <= K;
108
109                     WHEN K =>
110                         ImedOut <= '1';
111                         Rin(5) <= '1';
112                         prox <= L;
113
114                     WHEN L =>
115                         Rout(5) <= '1';
116                         Rin(indice) <= '1';
117                         prox <= A;
118
119             END CASE;
120     END IF;
121
122 END PROCESS;
123
124 PROCESS (Clock )
125 BEGIN
126         estado <= prox;
127 END PROCESS;
128
129
130
131
132 END UnidadeDeControleTeste;
```