



Sistemas Digitais Avançados

Prof. Imbiriba

Solução - Trabalho 2

Parte 1:

```
1  -- Display dígitos de 0 a 9 em display 7-segment, usando as chaves SW
2  -- como entradas.
3
4  LIBRARY ieee;
5  USE ieee.std_logic_1164.all;
6
7  ENTITY part1 IS
8  PORT ( SW          : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
9        LEDG         : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);    -- LEDs verdes
10       HEX1, HEX0    : OUT STD_LOGIC_VECTOR(0 TO 6));      -- 7-segs
11  END part1;
12
13  ARCHITECTURE Structure OF part1 IS
14  COMPONENT bcd7seg
15  PORT ( B : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
16        H : OUT STD_LOGIC_VECTOR(0 TO 6));
17  END COMPONENT;
18  BEGIN
19    LEDG <= SW;
20
21    -- saída para decodificação display 7-seg
22    digit1: bcd7seg PORT MAP (SW(7 DOWNTO 4), HEX1);
23    digit0: bcd7seg PORT MAP (SW(3 DOWNTO 0), HEX0);
24  END Structure;
25
26  LIBRARY ieee;
27  USE ieee.std_logic_1164.all;
28
29  ENTITY bcd7seg IS
30  PORT ( B : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
31        H : OUT STD_LOGIC_VECTOR(0 TO 6));
32  END bcd7seg;
33
34  ARCHITECTURE Structure OF bcd7seg IS
35  BEGIN
36  --
37  --      0
38  --      ---
39  --      | |
40  --      5| |1
41  --      | 6 |
42  --      ---
43  --      | |
44  --      4| |2
45  --      |  |
46  --      ---
47  --      3
48  --
49  -- B  H
50  -- -----
51  -- 0  0000001;
52  -- 1  1001111;
53  -- 2  0010010;
54  -- 3  0000110;
55  -- 4  1001100;
56  -- 5  0100100;
57  -- 6  0100000;
58  -- 7  0001111;
59  -- 8  0000000;
60  -- 9  0000100;
61  H(0) <= (NOT(B(3)) AND B(2) AND NOT(B(1)) AND NOT(B(0))) OR
62  (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)) AND B(0));
```



```
63 H(1) <= (B(2) AND NOT(B(1)) AND B(0)) OR
64      (B(2) AND B(1) AND NOT(B(0)));
65 H(2) <= (NOT(B(2)) AND B(1) AND NOT(B(0)));
66 H(3) <= (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)) AND B(0)) OR
67      (NOT(B(3)) AND B(2) AND NOT(B(1)) AND NOT(B(0)) OR (NOT(B(3)) AND B(2) AND B(1) AND B(0));
68 H(4) <= (NOT(B(1)) AND B(0)) OR (NOT(B(3)) AND B(0)) OR
69      (NOT(B(3)) AND B(2) AND NOT(B(1)));
70 H(5) <= (B(1) AND B(0)) OR (NOT(B(2)) AND B(1)) OR
71      (NOT(B(3)) AND NOT(B(2)) AND B(0));
72 H(6) <= (B(2) AND B(1) AND B(0)) OR (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)));
73 END Structure;
```

Parte 2:

```
1  -- conversor bcd-para-decimal
2
3  LIBRARY ieee;
4  USE ieee.std_logic_1164.all;
5
6  ENTITY part2 IS
7  PORT ( SW          : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
8        HEX1, HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6)); -- 7-segs
9  END part2;
10
11 ARCHITECTURE Structure OF part2 IS
12 COMPONENT mux2to1_4bit
13 PORT ( X, Y : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
14       s      : IN  STD_LOGIC;
15       M      : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
16 END COMPONENT;
17 COMPONENT bcd7seg
18 PORT ( B : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
19       H : OUT STD_LOGIC_VECTOR(0 TO 6));
20 END COMPONENT;
21 SIGNAL V, M : STD_LOGIC_VECTOR(3 DOWNTO 0);
22 SIGNAL A : STD_LOGIC_VECTOR(3 DOWNTO 0);
23 SIGNAL z : STD_LOGIC;
24 BEGIN
25   V <= SW;
26
27   -- circuito comparador para V > 9
28   z <= (V(3) AND V(2)) OR (V(3) AND V(1));
29
30   -- Circuito A: when V > 9, este circuito permite o display0 - d0 mostrar os valores
31   -- de 0 - 5 (para V = 10 até V = 15). Note que V3 = 1 para todos esses valores
32   -- e V3 não é necessário para o circuito A. O circuito implementa a seguinte tabela
33   -- verdade:
34   --
35   -- V2 V1 V0 | A2 A1 A0
```

Continua ...



```
36      -- -----
37      -- 0 1 0 | 0 0 0 (V = 1010 -> 0)
38      -- 0 1 1 | 0 0 1 (V = 1011 -> 1)
39      -- 1 0 0 | 0 1 0 (V = 1100 -> 2)
40      -- 1 0 1 | 0 1 1 (V = 1101 -> 3)
41      -- 1 1 0 | 1 0 0 (V = 1110 -> 4)
42      -- 1 1 1 | 1 0 1 (V = 1111 -> 5)
43      A(3) <= '0';
44      A(2) <= V(2) AND V(1);
45      A(1) <= V(2) AND NOT(V(1));
46      A(0) <= (V(1) AND V(0)) OR (V(2) AND V(0));
47
48      -- multiplexadores
49      U0: mux2tol_4bit PORT MAP (V, A, z, M);
50
51      U1: bcd7seg PORT MAP (M, HEX0);
52      U2: bcd7seg PORT MAP ("000" & z, HEX1);
53  END Structure;
54
55  LIBRARY ieee;
56  USE ieee.std_logic_1164.all;
57
58  -- Implementa um multiplexador de 4 bits de 2-para-1.
59  ENTITY mux2tol_4bit IS
60  PORT ( X, Y : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
61        s      : IN  STD_LOGIC;
62        M      : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
63  END mux2tol_4bit;
64
65  ARCHITECTURE Structure OF mux2tol_4bit IS
66  BEGIN
67      M(0) <= (NOT(s) AND X(0)) OR (s AND Y(0));
68      M(1) <= (NOT(s) AND X(1)) OR (s AND Y(1));
69      M(2) <= (NOT(s) AND X(2)) OR (s AND Y(2));
```

Continua ...



```
70 M(3) <= (NOT(s) AND X(3)) OR (s AND Y(3));
71 END Structure;
72
73 LIBRARY ieee;
74 USE ieee.std_logic_1164.all;
75
76 ENTITY bcd7seg IS
77 PORT ( B : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
78       H : OUT STD_LOGIC_VECTOR(0 TO 6));
79 END bcd7seg;
80
81 ARCHITECTURE Structure OF bcd7seg IS
82 BEGIN
83 --
84 --      0
85 --      ---
86 --      |  |
87 --      5|  |1
88 --      | 6 |
89 --      ---
90 --      |  |
91 --      4|  |2
92 --      |  |
93 --      ---
94 --      3
95 --
96 -- B H
97 -- -----
98 -- 0 0000001;
99 -- 1 1001111;
100 -- 2 0010010;
101 -- 3 0000110;
102 -- 4 1001100;
103 -- 5 0100100;
104 -- 6 0100000;
105 -- 7 0001111;
106 -- 8 0000000;
107 -- 9 0000100;
108 H(0) <= (NOT(B(3)) AND B(2) AND NOT(B(1)) AND NOT(B(0))) OR
109         (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)) AND B(0));
110 H(1) <= (B(2) AND NOT(B(1)) AND B(0)) OR
111         (B(2) AND B(1) AND NOT(B(0)));
112 H(2) <= (NOT(B(2)) AND B(1) AND NOT(B(0)));
113 H(3) <= (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)) AND B(0)) OR
114         (NOT(B(3)) AND B(2) AND NOT(B(1)) AND NOT(B(0))) OR (NOT(B(3)) AND B(2) AND B(1) AND B(0));
115 H(4) <= (NOT(B(1)) AND B(0)) OR (NOT(B(3)) AND B(0)) OR
116         (NOT(B(3)) AND B(2) AND NOT(B(1)));
117 H(5) <= (B(1) AND B(0)) OR (NOT(B(2)) AND B(1)) OR
118         (NOT(B(3)) AND NOT(B(2)) AND B(0));
119 H(6) <= (B(2) AND B(1) AND B(0)) OR (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)));
120 END Structure;
```



Parte 3:

```
1  -- somador completo de 4-bits
2
3  LIBRARY ieee;
4  USE ieee.std_logic_1164.all;
5
6  ENTITY part3 IS
7  PORT ( SW      : IN  STD_LOGIC_VECTOR(8 DOWNTO 0);
8        LEDG     : OUT STD_LOGIC_VECTOR(9 DOWNTO 0));
9  END part3;
10
11 ARCHITECTURE Structure OF part3 IS
12 COMPONENT fa
13 PORT ( a, b, ci : IN  STD_LOGIC;
14       s, co     : OUT STD_LOGIC);
15 END COMPONENT;
16 SIGNAL A, B, S : STD_LOGIC_VECTOR(3 DOWNTO 0);
17 SIGNAL C       : STD_LOGIC_VECTOR(4 DOWNTO 0);
18 BEGIN
19   A <= SW(7 DOWNTO 4);
20   B <= SW(3 DOWNTO 0);
21   C(0) <= SW(8);
22   bit0: fa PORT MAP (A(0), B(0), C(0), S(0), C(1));
23   bit1: fa PORT MAP (A(1), B(1), C(1), S(1), C(2));
24   bit2: fa PORT MAP (A(2), B(2), C(2), S(2), C(3));
25   bit3: fa PORT MAP (A(3), B(3), C(3), S(3), C(4));
26
27   -- Mostra as entradas
28   LEDG(4 DOWNTO 0) <= (C(4) & S);
29   LEDG(9 DOWNTO 5) <= "00000";
30 END Structure;
31
32
33 LIBRARY ieee;
34 USE ieee.std_logic_1164.all;
35
36 ENTITY fa IS
37 PORT ( a, b, ci : IN  STD_LOGIC;
38       s, co     : OUT STD_LOGIC);
39 END fa;
40
41 ARCHITECTURE Structure OF fa IS
42 SIGNAL a_xor_b : STD_LOGIC;
43 BEGIN
44   a_xor_b <= a XOR b;
45   s <= a_xor_b XOR ci;
46   co <= (NOT(a_xor_b) AND b) OR (a_xor_b AND ci);
47 END Structure;
48
```



Parte4:

```
1  -- Somador BCD de 1-digito S1 S0 = X + Y + Cin
2  -- entradas:SW7-4 = X
3  --           SW3-0 = Y
4  --           SW8 = Cin
5  -- Saídas: X é mostrado no HEX5
6  --           Y é mostrado no HEX3
7  --           S1 S0 é mostrado no HEX1 HEX0
8
9  LIBRARY ieee;
10 USE ieee.std_logic_1164.all;
11
12 ENTITY part4 IS
13 PORT ( SW : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
14       LEDG : OUT STD_LOGIC_VECTOR(9 DOWNTO 0);
15       HEX5, HEX4, HEX3, HEX2, HEX1, HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6));
16 END part4;
17
18 ARCHITECTURE Structure OF part4 IS
19 COMPONENT fa
20 PORT ( a, b, ci : IN STD_LOGIC;
21       s, co : OUT STD_LOGIC);
22 END COMPONENT;
23 COMPONENT bcd7seg
24 PORT ( B : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
25       H : OUT STD_LOGIC_VECTOR(0 TO 6));
26 END COMPONENT;
27 COMPONENT part2
28 PORT ( V : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
29       z : BUFFER STD_LOGIC;
30       M : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
31 END COMPONENT;
32 COMPONENT mux2to1_4bit
33 PORT ( X, Y : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
34       s : IN STD_LOGIC;
35       M : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
36 END COMPONENT;
37 SIGNAL X, Y, S : STD_LOGIC_VECTOR(3 DOWNTO 0); -- S é o resultado (saida) do somador
38 SIGNAL Cin : STD_LOGIC; -- carry de entrada
39 SIGNAL C : STD_LOGIC_VECTOR(4 DOWNTO 1); -- carries internos
40 SIGNAL M : STD_LOGIC_VECTOR(3 DOWNTO 0); -- usado para soma 0 até 15
41 SIGNAL B, S0 : STD_LOGIC_VECTOR(3 DOWNTO 0); -- usado para soma 16 - 19
42 SIGNAL z, S1 : STD_LOGIC;
43 BEGIN
44 X <= SW(7 DOWNTO 4);
45 Y <= SW(3 DOWNTO 0);
46 Cin <= SW(8);
47
48 bit0: fa PORT MAP (X(0), Y(0), Cin, S(0), C(1));
49 bit1: fa PORT MAP (X(1), Y(1), C(1), S(1), C(2));
50 bit2: fa PORT MAP (X(2), Y(2), C(2), S(2), C(3));
51 bit3: fa PORT MAP (X(3), Y(3), C(3), S(3), C(4));
52 LEDG(4 DOWNTO 0) <= (C(4) & S);
53
54 -- Display as entradas
55 H_5: bcd7seg PORT MAP (X, HEX5);
56 HEX4 <= "1111111"; -- display em branco (apagado)
57
58 H_3: bcd7seg PORT MAP (Y, HEX3);
59 HEX2 <= "1111111"; -- display em branco
60
61 -- Detecta entradas ilegais, mostra no LEDG(9)
62 LEDG(9) <= (X(3) AND X(2)) OR (X(3) AND X(1)) OR
63            (Y(3) AND Y(2)) OR (Y(3) AND Y(1));
64 LEDG(8 DOWNTO 5) <= "0000";
65
66 -- Mostra a soma
67 -- parte2 (V, z, M);
68 U1: part2 PORT MAP (S, z, M);
```



```
69     B(3) <= M(1);
70     B(2) <= NOT(M(1));
71     B(1) <= NOT(M(1));
72     B(0) <= M(0);
73     U2: mux2to1_4bit PORT MAP (M, B, C(4), S0);
74
75     H0: bcd7seg PORT MAP (S0, HEX0);
76     -- HEX1 mostra 1 quando z é 1 (soma 10-15), e também quando C[4] is 1 (soma 16-19)
77     S1 <= z OR C(4);
78     H1: bcd7seg PORT MAP ("000" & S1, HEX1);
79 END Structure;
80
81 LIBRARY ieee;
82 USE ieee.std_logic_1164.all;
83
84 ENTITY fa IS
85 PORT ( a, b, ci : IN STD_LOGIC;
86       s, co      : OUT STD_LOGIC);
87 END fa;
88
89 ARCHITECTURE Structure OF fa IS
90     SIGNAL a_xor_b : STD_LOGIC;
91 BEGIN
92     a_xor_b <= a XOR b;
93     s <= a_xor_b XOR ci;
94     co <= (NOT(a_xor_b) AND b) OR (a_xor_b AND ci);
95 END Structure;
96
97 LIBRARY ieee;
98 USE ieee.std_logic_1164.all;
99
100 ENTITY part2 IS
101 PORT ( V : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
102       z : BUFFER STD_LOGIC;
103       M : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
104 END part2;
105
106 ARCHITECTURE Structure OF part2 IS
107     SIGNAL A : STD_LOGIC_VECTOR(3 DOWNTO 0);
108     COMPONENT mux2to1_4bit
109     PORT ( X, Y : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
110          s      : IN STD_LOGIC;
111          M      : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
112     END COMPONENT;
113 BEGIN
114     -- circuito comparador para V > 9
115     z <= (V(3) AND V(2)) OR (V(3) AND V(1));
116
117     -- Circuito A: quando V > 9, este circuito permite o display d0 mostrar os valores
118     -- de 0 - 5 (para valores de V = 10 até V = 15). Note que V3 = 1 para todo esses
119     -- valores e V3 não é necessário no circuito A.
120     -- O circuito implementa a seguinte tabela verdade:
121     --
122     -- V2 V1 V0 | A2 A1 A0
123     -- -----
124     -- 0 1 0 | 0 0 0      V = 1010 -> 0
125     -- 0 1 1 | 0 0 1      V = 1011 -> 1
126     -- 1 0 0 | 0 1 0      V = 1100 -> 2
127     -- 1 0 1 | 0 1 1      V = 1101 -> 3
128     -- 1 1 0 | 1 0 0      V = 1110 -> 4
129     -- 1 1 1 | 1 0 1      V = 1111 -> 5
130     A(3) <= '0';
131     A(2) <= V(2) AND V(1);
132     A(1) <= V(2) AND NOT(V(1));
133     A(0) <= (V(1) AND V(0)) OR (V(2) AND V(0));
```

Continua ...



```
134
135     -- multiplexadores
136     U1: mux2tol_4bit PORT MAP (V, A, z, M);
137 END Structure;
138
139 LIBRARY ieee;
140 USE ieee.std_logic_1164.all;
141
142 -- Implementa um multiplexador de 4 bits de 2-para-1.
143 ENTITY mux2tol_4bit IS
144 PORT ( X, Y : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
145       s      : IN  STD_LOGIC;
146       M      : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
147 END mux2tol_4bit;
148
149 ARCHITECTURE Structure OF mux2tol_4bit IS
150 BEGIN
151     M(0) <= (NOT(s) AND X(0)) OR (s AND Y(0));
152     M(1) <= (NOT(s) AND X(1)) OR (s AND Y(1));
153     M(2) <= (NOT(s) AND X(2)) OR (s AND Y(2));
154     M(3) <= (NOT(s) AND X(3)) OR (s AND Y(3));
155 END Structure;
156
157 LIBRARY ieee;
158 USE ieee.std_logic_1164.all;
159
160 ENTITY bcd7seg IS
161 PORT ( B : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
162       H : OUT STD_LOGIC_VECTOR(0 TO 6));
163 END bcd7seg;
164
165 ARCHITECTURE Structure OF bcd7seg IS
166 BEGIN
167     --
168     --      0
169     --      ---
170     --      |   |
171     --      5|   |1
172     --      | 6 |
173     --      ---
174     --      |   |
175     --      4|   |2
176     --      |   |
177     --      ---
178     --      3
179     --
180     -- B  H
181     -- -----
182     -- 0  0000001;
183     -- 1  1001111;
184     -- 2  0010010;
185     -- 3  0000110;
186     -- 4  1001100;
187     -- 5  0100100;
188     -- 6  0100000;
189     -- 7  0001111;
190     -- 8  0000000;
191     -- 9  0000100;
192     H(0) <= (NOT(B(3)) AND B(2) AND NOT(B(1)) AND NOT(B(0))) OR
193             (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)) AND B(0));
194     H(1) <= (B(2) AND NOT(B(1)) AND B(0)) OR
195             (B(2) AND B(1) AND NOT(B(0)));
196     H(2) <= (NOT(B(2)) AND B(1) AND NOT(B(0)));
197     H(3) <= (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)) AND B(0)) OR
```

Continua ...



```
198         (NOT(B(3)) AND B(2) AND NOT(B(1)) AND NOT(B(0))) OR
199         (NOT(B(3)) AND B(2) AND B(1) AND B(0));
200     H(4) <= (NOT(B(1)) AND B(0)) OR (NOT(B(3)) AND B(0)) OR
201             (NOT(B(3)) AND B(2) AND NOT(B(1)));
202     H(5) <= (B(1) AND B(0)) OR (NOT(B(2)) AND B(1)) OR
203             (NOT(B(3)) AND NOT(B(2)) AND B(0));
204     H(6) <= (B(2) AND B(1) AND B(0)) OR (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)));
205 END Structure;
```

Parte5:

```
1  -- implementa um somador BCD de 2-dígitos S2 S1 S0 = A1 A0 + B1 B0
2  -- entradas: SW7-4 = A
3  --             SW3-0 = B
4  --             SW8 = Cin
5  -- saídas:   A é mostrado no display HEX5
6  --           B é mostrado no display HEX3
7  --           S1 S0 é mostrado no display HEX1 HEX0
8
9  LIBRARY ieee;
10 USE ieee.std_logic_1164.all;
11 USE ieee.std_logic_unsigned.all;
12
13 ENTITY part5 IS
14     PORT ( SW : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
15           HEX5, HEX4, HEX3, HEX2, HEX1, HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6));
16 END part5;
17
18 ARCHITECTURE Behavior OF part5 IS
19     COMPONENT bcd7seg
20     PORT ( bcd : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
21           display : OUT STD_LOGIC_VECTOR(0 TO 6));
22     END COMPONENT;
23
24     SIGNAL A, B : STD_LOGIC_VECTOR(3 DOWNTO 0);
25     SIGNAL Cin : STD_LOGIC;
26     SIGNAL S0 : STD_LOGIC_VECTOR(4 DOWNTO 0);
27     SIGNAL S1 : STD_LOGIC;
28     SIGNAL C1 : STD_LOGIC;
29
30     SIGNAL Z0 : STD_LOGIC_VECTOR(4 DOWNTO 0); -- usado para soma BCD
31     SIGNAL T0 : STD_LOGIC_VECTOR(4 DOWNTO 0); -- usado para soma BCD
32 BEGIN
33     A <= SW(7 DOWNTO 4);
34     B <= SW(3 DOWNTO 0);
35     Cin <= SW(8);
```

Continua ...



```
36
37 -- soma os dois digitos BCD mais baixos. Resultado possui 5 bits: C1,S0
38 T0 <= ('0' & A) + ('0' & B) + Cin;
39 PROCESS (T0)
40 BEGIN
41     IF (T0 > "01001") THEN
42         Z0 <= "01010"; -- necessário para subtrair 10 para conseguir o
43         C1 <= '1';      -- digito menos significativo
44     ELSE
45         Z0 <= "00000"; -- não é necessário subtrair algo quando soma <= 9
46         C1 <= '0';
47     END IF;
48 END PROCESS;
49 S0 <= T0 - Z0; -- subtract 10 ou 0
50 S1 <= C1;
51
52 -- mostra nos displays de 7-seg
53 digit3: bcd7seg PORT MAP (A, HEX5);
54 digit2: bcd7seg PORT MAP (B, HEX3);
55 digit1: bcd7seg PORT MAP ("000" & S1, HEX1);
56 digit0: bcd7seg PORT MAP (S0(3 DOWNT0 0), HEX0);
57
58 HEX4 <= "1111111";
59 HEX2 <= "1111111";
60 END Behavior;
61
62 LIBRARY ieee;
63 USE ieee.std_logic_1164.all;
64
65 ENTITY bcd7seg IS
66     PORT ( bcd : IN STD_LOGIC_VECTOR(3 DOWNT0 0);
67           display : OUT STD_LOGIC_VECTOR(0 TO 6));
68 END bcd7seg;
69
70 ARCHITECTURE Behavior OF bcd7seg IS
71 BEGIN
72     --
73     --      0
74     --      ---
75     --      |   |
76     --      5|   |1
77     --      | 6 |
78     --      ---
79     --      |   |
80     --      4|   |2
81     --      |   |
82     --      ---
83     --      3
84     --
85     PROCESS (bcd)
86     BEGIN
87         CASE bcd IS
88             WHEN "0000" => display <= "0000001";
89             WHEN "0001" => display <= "1001111";
90             WHEN "0010" => display <= "0010010";
91             WHEN "0011" => display <= "0000110";
92             WHEN "0100" => display <= "1001100";
93             WHEN "0101" => display <= "0100100";
94             WHEN "0110" => display <= "0100000";
95             WHEN "0111" => display <= "0001111";
96             WHEN "1000" => display <= "0000000";
97             WHEN "1001" => display <= "0000100";
98             WHEN OTHERS => display <= "1111111";
99         END CASE;
100     END PROCESS;
101 END Behavior;
```



Parte6:

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.std_logic_unsigned.all;
4
5  -- entrada de 6-bits usando as chaves SW, e converte para decimal (2-digitos bcd)
6  ENTITY part6 IS
7  PORT ( SW : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
8        HEX3, HEX2, HEX1, HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6)); -- 7-segmentos
9  END part6;
10
11 ARCHITECTURE Behavior OF part6 IS
12 COMPONENT bcd7seg
13 PORT ( bcd : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
14       display : OUT STD_LOGIC_VECTOR(0 TO 6));
15 END COMPONENT;
16
17 SIGNAL bcd_h : STD_LOGIC_VECTOR(3 DOWNTO 0);
18 SIGNAL bin6, bcd_l : STD_LOGIC_VECTOR(5 DOWNTO 0);
19 BEGIN
20     bin6 <= SW;
21
22     -- Check various ranges and ajusta para digits bcd.
23     PROCESS (bin6)
24     BEGIN
25         IF (bin6 < "001010") THEN
26             bcd_h <= "0000";
27             bcd_l <= bin6;
28         ELSIF (bin6 < "010100") THEN
29             bcd_h <= "0001";
30             bcd_l <= bin6 - "001010";
31         ELSIF (bin6 < "011110") THEN
32             bcd_h <= "0010";
33             bcd_l <= bin6 - "010100";
34         ELSIF (bin6 < "101000") THEN
35             bcd_h <= "0011";
36             bcd_l <= bin6 - "011110";
37         ELSIF (bin6 < "110010") THEN
38             bcd_h <= "0100";
39             bcd_l <= bin6 - "101000";
40         ELSIF (bin6 < "111100") THEN
41             bcd_h <= "0101";
42             bcd_l <= bin6 - "110010";
43         ELSE
44             bcd_h <= "0110";
45             bcd_l <= bin6 - "111100";
46         END IF;
47     END PROCESS;
48
49     -- alimenta os displays
50     digit1: bcd7seg PORT MAP (bcd_h, HEX1);
51     digit0: bcd7seg PORT MAP (bcd_l(3 DOWNTO 0), HEX0);
52
53     -- apaga os displays adjacentes
54     digit3: bcd7seg PORT MAP ("1111", HEX3);
55     digit2: bcd7seg PORT MAP ("1111", HEX2);
56
57 END Behavior;
58
59 LIBRARY ieee;
60 USE ieee.std_logic_1164.all;
61
62 ENTITY bcd7seg IS
63 PORT ( bcd : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
64       display : OUT STD_LOGIC_VECTOR(0 TO 6));
65 END bcd7seg;
```

Continua ...



```
67 ARCHITECTURE Behavior OF bcd7seg IS
68 BEGIN
69     --
70     --      0
71     --    ---
72     --    |  |
73     --   5|  |1
74     --    | 6 |
75     --    ---
76     --    |  |
77     --   4|  |2
78     --    |  |
79     --    ---
80     --      3
81     --
82     PROCESS (bcd)
83     BEGIN
84         CASE bcd IS
85             WHEN "0000" => display <= "0000001";
86             WHEN "0001" => display <= "1001111";
87             WHEN "0010" => display <= "0010010";
88             WHEN "0011" => display <= "0000110";
89             WHEN "0100" => display <= "1001100";
90             WHEN "0101" => display <= "0100100";
91             WHEN "0110" => display <= "0100000";
92             WHEN "0111" => display <= "0001111";
93             WHEN "1000" => display <= "0000000";
94             WHEN "1001" => display <= "0000100";
95             WHEN OTHERS => display <= "1111111";
96         END CASE;
97     END PROCESS;
98 END Behavior;
```