```
1   library ieee ;
2   use ieee.std_logic_1164.all;
3
4   entity projeto02 is
5      port ( clock_27,SW1,SW0,KEY3: in std_logic ;
6             LEDR1,LEDR0: out std_logic;
7             P02HEX5,P02HEX4, P02HEX3, P02HEX2, P02HEX1, P02HEX0: out std_logic_vector(6
    downto 0);
8             est_maq_fifo: out std_logic_vector(2 downto 0));
9   end projeto02;
10
11  architecture ckt of projeto02 is
12     -- Divisor de Clock
13     component divisorClock is
14        port (clk_in : in std_logic ;
15              clk_out : out std_logic );
16     end component;
17
18     -- ROM
19     component romFIFO is
20        PORT (address: IN STD_LOGIC_VECTOR (5 DOWNTO 0);
21              clock: IN STD_LOGIC  := '1';
22              q: OUT STD_LOGIC_VECTOR (12 DOWNTO 0));
23     end component;
24
25     --FIFO
26     component fifo is
27        port (CLK_fifo, WR, RD, reset: in std_logic;
28              W_data: in std_logic_vector(12 downto 0);
29              em,fu: out std_logic;
30              R_data: out std_logic_vector(12 downto 0);
31              Estados_maquina: out std_logic_vector(2 downto 0));
32     end component;
33
34     --Contador
35     component contador6Bits is
36       port ( clk_c6B,ld_c6B,clr_c6B: in std_logic;
37              out_c6B: out std_logic_vector(5 downto 0));
38     end component;
39
40     --BINBCDtoD7SEG
41     component bitToBcd13bitsToD7Seg is
42       port (SW_in: in std_logic_vector(12 downto 0);
43             HEX0,HEX1,HEX2,HEX3: out std_logic_vector(6 downto 0));
44     end component;
45
46     signal CLK_1, led_full, led_empty, clr_reverso: std_logic;
47     signal saida_rom, saida_fifo: std_logic_vector(12 downto 0);
48     signal saida_contador: std_logic_vector(5 downto 0);
49     signal lixo1,lixo0, hx00,hx01,hx02,hx03,hx04,hx05: std_logic_vector(6 downto 0);
50     signal estados: std_logic_vector(2 downto 0);
51
52     begin
53
54        clr_reverso <= not KEY3;
55
56        DClock: divisorClock port map(
57           clk_in => clock_27,
58           clk_out => CLK_1);
59
60        FIFO16x13: fifo port map(
61           CLK_fifo => CLK_1,
62           WR => SW1,
63           RD => SW0,
64           reset => clr_reverso,
65           W_data => saida_rom,
```

```
66            em => led_empty,
67            fu => led_full,
68            R_data => saida_fifo,
69            Estados_maquina => estados);
70
71         ROM: romFIFO port map(
72            address => saida_contador,
73            clock => CLK_1,
74            q => saida_rom);
75
76         Contador: contador6Bits port map(
77            clk_c6B => CLK_1,
78            ld_c6B => SW1,
79            clr_c6B => clr_reverso,
80            out_c6B => saida_contador);
81
82         DisplayRead: bitToBcd13bitsToD7Seg port map(
83            SW_in => saida_fifo,
84            HEX0 => hx00,
85            HEX1 => hx01,
86            HEX2 => hx02,
87            HEX3 => hx03);
88
89         DisplayContador: bitToBcd13bitsToD7Seg port map(
90            SW_in(12 downto 6) =>"0000000",
91            SW_in(5 downto 0) => saida_contador,
92            HEX0 => hx04,
93            HEX1 => hx05,
94            HEX2 => lixo0,
95            HEX3 => lixo1);
96
97         -- HEX
98         P02HEX0 <= hx00;
99         P02HEX1 <= hx01;
100        P02HEX2 <= hx02;
101        P02HEX3 <= hx03;
102        P02HEX4 <= hx04;
103        P02HEX5 <= hx05;
104
105        -- Estados da Maquina FIFO
106        est_maq_fifo <= estados;
107
108        -- Luzes de aviso
109        LEDR0 <= led_full;
110        LEDR1 <= led_empty;
111  end ckt;
```

```
1   library ieee ;
2   use ieee.std_logic_1164.all;
3   entity divisorClock is
4      port ( clk_in : in std_logic ;
5             clk_out : out std_logic );
6   end divisorClock ;
7   architecture ckt of divisorClock is
8      signal ax : std_logic ;
9      begin
10        process ( clk_in )
11           variable cnt: integer range 0 to 13500000 := 0;
12           begin
13              if ( rising_edge ( clk_in )) then
14                 if (cnt =13500000) then
15                    cnt := 0;
16                    ax <= not ax;
17                 else
18                    cnt := cnt +1;
19                 end if;
20              end if;
21        end process ;
22        clk_out <= ax;
23  end ckt;
```

```
1  -- megafunction wizard: %ROM: 1-PORT%
2  -- GENERATION: STANDARD
3  -- VERSION: WM1.0
4  -- MODULE: altsyncram
5
6  -- ============================================================
7  -- File Name: romFIFO.vhd
8  -- Megafunction Name(s):
9  --          altsyncram
10 --
11 -- Simulation Library Files(s):
12 --          altera_mf
13 -- ============================================================
14 -- ************************************************************
15 -- THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
16 --
17 -- 13.0.0 Build 156 04/24/2013 SJ Web Edition
18 -- ************************************************************
19
20
21 --Copyright (C) 1991-2013 Altera Corporation
22 --Your use of Altera Corporation's design tools, logic functions
23 --and other software and tools, and its AMPP partner logic
24 --functions, and any output files from any of the foregoing
25 --(including device programming or simulation files), and any
26 --associated documentation or information are expressly subject
27 --to the terms and conditions of the Altera Program License
28 --Subscription Agreement, Altera MegaCore Function License
29 --Agreement, or other applicable license agreement, including,
30 --without limitation, that your use is for the sole purpose of
31 --programming logic devices manufactured by Altera and sold by
32 --Altera or its authorized distributors.  Please refer to the
33 --applicable agreement for further details.
34
35
36 LIBRARY ieee;
37 USE ieee.std_logic_1164.all;
38
39 LIBRARY altera_mf;
40 USE altera_mf.all;
41
42 ENTITY romFIFO IS
43    PORT
44    (
45       address     : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
46       clock    : IN STD_LOGIC  := '1';
47       q     : OUT STD_LOGIC_VECTOR (12 DOWNTO 0)
48    );
49 END romFIFO;
50
51
52 ARCHITECTURE SYN OF romfifo IS
53
54    SIGNAL sub_wire0  : STD_LOGIC_VECTOR (12 DOWNTO 0);
55
56
57
58    COMPONENT altsyncram
59    GENERIC (
60       clock_enable_input_a    : STRING;
61       clock_enable_output_a      : STRING;
62       init_file      : STRING;
63       intended_device_family     : STRING;
64       lpm_hint    : STRING;
65       lpm_type    : STRING;
66       numwords_a     : NATURAL;
```

```
67          operation_mode    : STRING;
68          outdata_aclr_a    : STRING;
69          outdata_reg_a     : STRING;
70          widthad_a       : NATURAL;
71          width_a     : NATURAL;
72          width_byteena_a       : NATURAL
73       );
74       PORT (
75             address_a   : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
76             clock0   : IN STD_LOGIC ;
77             q_a   : OUT STD_LOGIC_VECTOR (12 DOWNTO 0)
78       );
79       END COMPONENT;
80
81    BEGIN
82       q    <= sub_wire0(12 DOWNTO 0);
83
84       altsyncram_component : altsyncram
85       GENERIC MAP (
86          clock_enable_input_a => "BYPASS",
87          clock_enable_output_a => "BYPASS",
88          init_file => "romFIFO.mif",
89          intended_device_family => "Cyclone II",
90          lpm_hint => "ENABLE_RUNTIME_MOD=NO",
91          lpm_type => "altsyncram",
92          numwords_a => 64,
93          operation_mode => "ROM",
94          outdata_aclr_a => "NONE",
95          outdata_reg_a => "UNREGISTERED",
96          widthad_a => 6,
97          width_a => 13,
98          width_byteena_a => 1
99       )
100      PORT MAP (
101         address_a => address,
102         clock0 => clock,
103         q_a => sub_wire0
104      );
105
106
107
108   END SYN;
109
110   -- ============================================================
111   -- CNX file retrieval info
112   -- ============================================================
113   -- Retrieval info: PRIVATE: ADDRESSSTALL_A NUMERIC "0"
114   -- Retrieval info: PRIVATE: AclrAddr NUMERIC "0"
115   -- Retrieval info: PRIVATE: AclrByte NUMERIC "0"
116   -- Retrieval info: PRIVATE: AclrOutput NUMERIC "0"
117   -- Retrieval info: PRIVATE: BYTE_ENABLE NUMERIC "0"
118   -- Retrieval info: PRIVATE: BYTE_SIZE NUMERIC "8"
119   -- Retrieval info: PRIVATE: BlankMemory NUMERIC "0"
120   -- Retrieval info: PRIVATE: CLOCK_ENABLE_INPUT_A NUMERIC "0"
121   -- Retrieval info: PRIVATE: CLOCK_ENABLE_OUTPUT_A NUMERIC "0"
122   -- Retrieval info: PRIVATE: Clken NUMERIC "0"
123   -- Retrieval info: PRIVATE: IMPLEMENT_IN_LES NUMERIC "0"
124   -- Retrieval info: PRIVATE: INIT_FILE_LAYOUT STRING "PORT_A"
125   -- Retrieval info: PRIVATE: INIT_TO_SIM_X NUMERIC "0"
126   -- Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING "Cyclone II"
127   -- Retrieval info: PRIVATE: JTAG_ENABLED NUMERIC "0"
128   -- Retrieval info: PRIVATE: JTAG_ID STRING "NONE"
129   -- Retrieval info: PRIVATE: MAXIMUM_DEPTH NUMERIC "0"
130   -- Retrieval info: PRIVATE: MIFfilename STRING "romFIFO.mif"
131   -- Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "64"
132   -- Retrieval info: PRIVATE: RAM_BLOCK_TYPE NUMERIC "0"
```

```
133   -- Retrieval info: PRIVATE: RegAddr NUMERIC "1"
134   -- Retrieval info: PRIVATE: RegOutput NUMERIC "0"
135   -- Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING "0"
136   -- Retrieval info: PRIVATE: SingleClock NUMERIC "1"
137   -- Retrieval info: PRIVATE: UseDQRAM NUMERIC "0"
138   -- Retrieval info: PRIVATE: WidthAddr NUMERIC "6"
139   -- Retrieval info: PRIVATE: WidthData NUMERIC "13"
140   -- Retrieval info: PRIVATE: rden NUMERIC "0"
141   -- Retrieval info: LIBRARY: altera_mf altera_mf.altera_mf_components.all
142   -- Retrieval info: CONSTANT: CLOCK_ENABLE_INPUT_A STRING "BYPASS"
143   -- Retrieval info: CONSTANT: CLOCK_ENABLE_OUTPUT_A STRING "BYPASS"
144   -- Retrieval info: CONSTANT: INIT_FILE STRING "romFIFO.mif"
145   -- Retrieval info: CONSTANT: INTENDED_DEVICE_FAMILY STRING "Cyclone II"
146   -- Retrieval info: CONSTANT: LPM_HINT STRING "ENABLE_RUNTIME_MOD=NO"
147   -- Retrieval info: CONSTANT: LPM_TYPE STRING "altsyncram"
148   -- Retrieval info: CONSTANT: NUMWORDS_A NUMERIC "64"
149   -- Retrieval info: CONSTANT: OPERATION_MODE STRING "ROM"
150   -- Retrieval info: CONSTANT: OUTDATA_ACLR_A STRING "NONE"
151   -- Retrieval info: CONSTANT: OUTDATA_REG_A STRING "UNREGISTERED"
152   -- Retrieval info: CONSTANT: WIDTHAD_A NUMERIC "6"
153   -- Retrieval info: CONSTANT: WIDTH_A NUMERIC "13"
154   -- Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"
155   -- Retrieval info: USED_PORT: address 0 0 6 0 INPUT NODEFVAL "address[5..0]"
156   -- Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"
157   -- Retrieval info: USED_PORT: q 0 0 13 0 OUTPUT NODEFVAL "q[12..0]"
158   -- Retrieval info: CONNECT: @address_a 0 0 6 0 address 0 0 6 0
159   -- Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0
160   -- Retrieval info: CONNECT: q 0 0 13 0 @q_a 0 0 13 0
161   -- Retrieval info: GEN_FILE: TYPE_NORMAL romFIFO.vhd TRUE
162   -- Retrieval info: GEN_FILE: TYPE_NORMAL romFIFO.inc FALSE
163   -- Retrieval info: GEN_FILE: TYPE_NORMAL romFIFO.cmp FALSE
164   -- Retrieval info: GEN_FILE: TYPE_NORMAL romFIFO.bsf FALSE
165   -- Retrieval info: GEN_FILE: TYPE_NORMAL romFIFO_inst.vhd FALSE
166   -- Retrieval info: LIB_FILE: altera_mf
167
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity contador6Bits is
5     port ( clk_c6B,ld_c6B,clr_c6B: in std_logic;
6            out_c6B: out std_logic_vector(5 downto 0));
7   end contador6Bits ;
8   architecture ckt of contador6Bits is
9      component ffjk is
10        port (clk ,J ,K ,P ,C: in std_logic;
11              q: out std_logic);
12     end component;
13
14     signal q0_out,q1_out,q2_out,q3_out,q4_out,q5_out, clr_reverso: std_logic;
15     signal resp_and: std_logic_vector(4 downto 0);
16
17     begin
18
19        clr_reverso <= not clr_c6B;
20        Q0: ffjk port map(
21             clk => clk_c6B,
22             J => ld_c6B,
23             K => ld_c6B,
24             P => '1',
25             C => clr_reverso,
26             q => q0_out);
27
28        resp_and(0) <= q0_out and ld_c6B;
29
30        Q1: ffjk port map(
31             clk => clk_c6B,
32             J => resp_and(0),
33             K => resp_and(0),
34             P => '1',
35             C => clr_reverso,
36             q => q1_out);
37
38        resp_and(1) <= q1_out and resp_and(0);
39
40        Q2: ffjk port map(
41             clk => clk_c6B,
42             J => resp_and(1),
43             K => resp_and(1),
44             P => '1',
45             C => clr_reverso,
46             q => q2_out);
47
48        resp_and(2) <= q2_out and resp_and(1);
49
50        Q3: ffjk port map(
51             clk => clk_c6B,
52             J => resp_and(2),
53             K => resp_and(2),
54             P => '1',
55             C => clr_reverso,
56             q => q3_out);
57
58        resp_and(3) <= q3_out and resp_and(2);
59
60        Q4: ffjk port map(
61             clk => clk_c6B,
62             J => resp_and(3),
63             K => resp_and(3),
64             P => '1',
65             C => clr_reverso,
66             q => q4_out);
```

```
67
68          resp_and(4) <= q4_out and resp_and(3);
69
70          Q5: ffjk port map(
71               clk => clk_c6B,
72               J => resp_and(4),
73               K => resp_and(4),
74               P => '1',
75               C => clr_reverso,
76               q => q5_out);
77
78          out_c6B(0) <= q0_out;
79          out_c6B(1) <= q1_out;
80          out_c6B(2) <= q2_out;
81          out_c6B(3) <= q3_out;
82          out_c6B(4) <= q4_out;
83          out_c6B(5) <= q5_out;
84
85
86    end ckt ;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   ENTITY ffjk IS
5     port ( clk ,J ,K ,P ,C : in std_logic;
6              q : out std_logic );
7   END ffjk ;
8   ARCHITECTURE ckt OF ffjk IS
9   SIGNAL qS : std_logic;
10  BEGIN
11    PROCESS ( clk ,P ,C )
12    BEGIN
13      IF P = '0' THEN qS <= '1';
14      ELSIF C ='0' THEN qS <= '0';
15      ELSIF clk = '1' AND clk ' EVENT THEN
16        IF J = '1' AND K = '1' THEN qS <= NOT qS ;
17        ELSIF J = '1' AND K = '0' THEN qS <= '1';
18        ELSIF J = '0' AND K = '1' THEN qS <= '0';
19        END IF;
20      END IF;
21    END PROCESS ;
22  q <= qS ;
23  END ckt ;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity bitToBcd13bitsToD7Seg is
5     port (SW_in: in std_logic_vector(12 downto 0);
6           HEX0,HEX1,HEX2,HEX3: out std_logic_vector(6 downto 0));
7   end bitToBcd13bitsToD7Seg ;
8
9   architecture ckt of bitToBcd13bitsToD7Seg is
10    component bitToBcd13bits is
11      port (bit_in: in std_logic_vector(12 downto 0);
12            bcd_out: out std_logic_vector(15 downto 0));
13    end component;
14    component d7Seg is
15       port(S_in: in std_logic_vector(3 downto 0);
16            D_out: out std_logic_vector(6 downto 0));
17    end component;
18
19    signal bcd_ax_out: std_logic_vector(15 downto 0);
20    signal hex0_ax_out,hex1_ax_out,hex2_ax_out,hex3_ax_out : std_logic_vector(6 downto 0);
21
22    begin
23        BtB: BitToBcd13bits port map(
24             bit_in => SW_in,
25             bcd_out => bcd_ax_out);
26
27        D7S3: d7Seg port map(
28              S_in => bcd_ax_out(15 downto 12),
29              D_out => hex3_ax_out);
30
31        D7S2: d7Seg port map(
32              S_in => bcd_ax_out(11 downto 8),
33              D_out => hex2_ax_out);
34
35        D7S1: d7Seg port map(
36              S_in => bcd_ax_out(7 downto 4),
37              D_out => hex1_ax_out);
38
39        D7S0: d7Seg port map(
40              S_in => bcd_ax_out(3 downto 0),
41              D_out => hex0_ax_out);
42        HEX0 <= hex0_ax_out;
43        HEX1 <= hex1_ax_out;
44        HEX2 <= hex2_ax_out;
45        HEX3 <= hex3_ax_out;
46  end ckt;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity d7Seg is
5     port(S_in: in std_logic_vector(3 downto 0);
6          D_out: out std_logic_vector(6 downto 0));
7   end d7Seg;
8
9   architecture display of d7Seg is
10    begin
11      D_out(0) <= not ((S_in(3) or S_in(1) or (S_in(2) and S_in(0)) or (S_in(2) nor S_in(0
    ))));
12
13      D_out(1) <= not (((not S_in(2)) or (S_in(1) nor S_in(0)) or (S_in(1) and S_in(0))));
14
15      D_out(2) <= not (((not(S_in(3)) and ((S_in(1) and S_in(0)) or S_in(2))) or (not(S_in(2
    ) or S_in(1)))));
16
17      D_out(3) <= not((S_in(3) or (S_in(1) and (S_in(2) nand S_in(0))) or (S_in(2) nor S_in(
    0)) or (S_in(2) and (not S_in(1)) and S_in(0))));
18
19      D_out(4) <= not (((S_in(1) and (not S_in(0))) or (S_in(2) nor S_in(0))));
20
21      D_out(5) <= not ((S_in(3) or (S_in(1) nor S_in(0)) or (S_in(2) and (S_in(1) nand S_in(
    0)))));
22
23      D_out(6) <= not ((S_in(3) or (S_in(1) and (not S_in(2))) or (S_in(2) and (S_in(1) nand
     S_in(0)))));
24
25  end display;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity bitToBcd13bits is
5     port (bit_in: in std_logic_vector(12 downto 0);
6           bcd_out: out std_logic_vector(15 downto 0));
7   end bitToBcd13bits ;
8
9   architecture ckt of bitToBcd13bits is
10    component ciBitToBcd is
11      port (BtB_in: in std_logic_vector(3 downto 0);
12            BtB_out: out std_logic_vector(3 downto 0));
13    end component;
14
15    signal ciBtB_01_out, ciBtB_02_out, ciBtB_03_out, ciBtB_04_out, ciBtB_05_out,
    ciBtB_06_out, ciBtB_07_out, ciBtB_08_out, ciBtB_09_out: std_logic_vector(3 downto 0);
16    signal ciBtB_10_out, ciBtB_11_out, ciBtB_12_out, ciBtB_13_out, ciBtB_14_out,
    ciBtB_15_out, ciBtB_16_out, ciBtB_17_out, ciBtB_18_out: std_logic_vector(3 downto 0);
17    signal ciBtB_19_out, ciBtB_20_out, ciBtB_21_out: std_logic_vector(3 downto 0);
18
19    begin
20       ciBtB01: ciBitToBcd port map(
21                BtB_in(3) => '0',
22                BtB_in(2 downto 0) => bit_in(12 downto 10),
23                BtB_out => ciBtB_01_out);
24
25       ciBtB02: ciBitToBcd port map(
26                BtB_in(3 downto 1) => ciBtB_01_out(2 downto 0),
27                BtB_in(0) => bit_in(9),
28                BtB_out => ciBtB_02_out);
29
30       ciBtB03: ciBitToBcd port map(
31                BtB_in(3 downto 1) => ciBtB_02_out(2 downto 0),
32                BtB_in(0) => bit_in(8),
33                BtB_out => ciBtB_03_out);
34
35       ciBtB04: ciBitToBcd port map(
36                BtB_in(3) => '0',
37                BtB_in(2) => ciBtB_01_out(3),
38                BtB_in(1) => ciBtB_02_out(3),
39                BtB_in(0) => ciBtB_03_out(3),
40                BtB_out => ciBtB_04_out);
41
42       ciBtB05: ciBitToBcd port map(
43                BtB_in(3 downto 1)=> ciBtB_03_out(2 downto 0),
44                BtB_in(0) => bit_in(7),
45                BtB_out => ciBtB_05_out);
46
47       ciBtB06: ciBitToBcd port map(
48                BtB_in(3 downto 1)=> ciBtB_04_out(2 downto 0),
49                BtB_in(0) => ciBtB_05_out(3),
50                BtB_out => ciBtB_06_out);
51
52       ciBtB07: ciBitToBcd port map(
53                BtB_in(3 downto 1)=> ciBtB_05_out(2 downto 0),
54                BtB_in(0) => bit_in(6),
55                BtB_out => ciBtB_07_out);
56
57       ciBtB08: ciBitToBcd port map(
58                BtB_in(3 downto 1)=> ciBtB_06_out(2 downto 0),
59                BtB_in(0) => ciBtB_07_out(3),
60                BtB_out => ciBtB_08_out);
61
62       ciBtB09: ciBitToBcd port map(
63                BtB_in(3 downto 1)=> ciBtB_07_out(2 downto 0),
64                BtB_in(0) => bit_in(5),
```

```
65                BtB_out => ciBtB_09_out);
66
67        ciBtB10: ciBitToBcd port map(
68                BtB_in(3) => '0',
69                BtB_in(2) => ciBtB_04_out(3),
70                BtB_in(1) => ciBtB_06_out(3),
71                BtB_in(0) => ciBtB_08_out(3),
72                BtB_out => ciBtB_10_out);
73
74        ciBtB11: ciBitToBcd port map(
75                BtB_in(3 downto 1)=> ciBtB_08_out(2 downto 0),
76                BtB_in(0) => ciBtB_09_out(3),
77                BtB_out => ciBtB_11_out);
78
79        ciBtB12: ciBitToBcd port map(
80                BtB_in(3 downto 1)=> ciBtB_09_out(2 downto 0),
81                BtB_in(0) => bit_in(4),
82                BtB_out => ciBtB_12_out);
83
84        ciBtB13: ciBitToBcd port map(
85                BtB_in(3 downto 1)=> ciBtB_10_out(2 downto 0),
86                BtB_in(0) => ciBtB_11_out(3),
87                BtB_out => ciBtB_13_out);
88
89        ciBtB14: ciBitToBcd port map(
90                BtB_in(3 downto 1)=> ciBtB_11_out(2 downto 0),
91                BtB_in(0) => ciBtB_12_out(3),
92                BtB_out => ciBtB_14_out);
93
94        ciBtB15: ciBitToBcd port map(
95                BtB_in(3 downto 1)=> ciBtB_12_out(2 downto 0),
96                BtB_in(0) => bit_in(3),
97                BtB_out => ciBtB_15_out);
98
99        ciBtB16: ciBitToBcd port map(
100               BtB_in(3 downto 1)=> ciBtB_13_out(2 downto 0),
101               BtB_in(0) => ciBtB_14_out(3),
102               BtB_out => ciBtB_16_out);
103
104       ciBtB17: ciBitToBcd port map(
105               BtB_in(3 downto 1)=> ciBtB_14_out(2 downto 0),
106               BtB_in(0) => ciBtB_15_out(3),
107               BtB_out => ciBtB_17_out);
108
109       ciBtB18: ciBitToBcd port map(
110               BtB_in(3 downto 1)=> ciBtB_15_out(2 downto 0),
111               BtB_in(0) => bit_in(2),
112               BtB_out => ciBtB_18_out);
113
114       ciBtB19: ciBitToBcd port map(
115               BtB_in(3 downto 1)=> ciBtB_16_out(2 downto 0),
116               BtB_in(0) => ciBtB_17_out(3),
117               BtB_out => ciBtB_19_out);
118
119       ciBtB20: ciBitToBcd port map(
120               BtB_in(3 downto 1)=> ciBtB_17_out(2 downto 0),
121               BtB_in(0) => ciBtB_18_out(3),
122               BtB_out => ciBtB_20_out);
123
124       ciBtB21: ciBitToBcd port map(
125               BtB_in(3 downto 1)=> ciBtB_18_out(2 downto 0),
126               BtB_in(0) => bit_in(1),
127               BtB_out => ciBtB_21_out);
128
129       bcd_out(15) <= ciBtB_10_out(3);
130       bcd_out(14) <= ciBtB_13_out(3);
```

```
131         bcd_out(13) <= ciBtB_16_out(3);
132         bcd_out(12 downto 9) <= ciBtB_19_out;
133         bcd_out(8 downto 5) <= ciBtB_20_out;
134         bcd_out(4 downto 1) <= ciBtB_21_out;
135         bcd_out(0) <= bit_in(0);
136 
137     end ckt;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity ciBitToBcd is
5     port (BtB_in: in bit_vector(3 downto 0);
6           BtB_out: out bit_vector(3 downto 0));
7   end ciBitToBcd;
8
9   architecture ckt of ciBitToBcd is
10  begin
11    BtB_out(3) <= (BtB_in(3) or (BtB_in(2) and (BtB_in(1) or BtB_in(0))));
12    BtB_out(2)<= ((BtB_in(2) and (not BtB_in(1)) and (not BtB_in(0))) or (BtB_in(3) and
    BtB_in(0)));
13    BtB_out(1)<= ((BtB_in(3) and (not BtB_in(0))) or (BtB_in(1) and ((not BtB_in(2)) or
    BtB_in(0))));
14    BtB_out(0)<= (((not BtB_in(3)) and (not BtB_in(2)) and BtB_in(0)) or ((not BtB_in(0))
    and (BtB_in(3) or (BtB_in(2) and BtB_in(1)))));
15  end ckt;
16
```

```
1   library ieee ;
2   use ieee.std_logic_1164.all;
3
4   entity fifo is
5      port (CLK_fifo, WR, RD, reset: in std_logic;
6            W_data: in std_logic_vector(12 downto 0);
7            em,fu: out std_logic;
8            R_data: out std_logic_vector(12 downto 0);
9            Estados_maquina: out std_logic_vector(2 downto 0));
10  end fifo;
11
12  architecture ckt of fifo is
13
14     component blocoDeControleFIFO is
15        port (clk_ff, rst_ff, rd_ff, wr_ff: in std_logic;
16              equal_cont_wd_e_rd, equal_cont_wd_1_e_rd, equal_cont_wd_e_rd_1: in std_logic;
17              led_vazio_ff, led_cheio_ff,load_wr,load_rd,clr_ff: out std_logic;
18              saida_mv_fifo: out std_logic_vector(2 downto 0));
19     end component;
20
21     component datapathFIFO is
22       port (wr_data_dp: in std_logic_vector(12 downto 0);
23             ld_wr_dp, ld_rd_dp, clr_fifo_dp, clk_fifo_dp: in std_logic;
24             eq_comp_wr_rd,eq_comp_wr1_rd,eq_comp_wr_rd1: out std_logic;
25             rd_data_dp: out std_logic_vector(12 downto 0));
26     end component;
27
28
29     signal leitura_banco_registradores: std_logic_vector(12 downto 0);
30     signal l_vazio, l_cheio, saida_load_wr, saida_load_rd, clr_dt : std_logic;
31     signal saida_eq_cont_wr_rd, saida_eq_cont_wr1_rd, saida_eq_cont_wr_rd1: std_logic;
32     signal saida_fifo: std_logic_vector(2 downto 0);
33
34     begin
35        BlocodeControle: blocoDeControleFIFO port map(
36           clk_ff => CLK_fifo,
37           rst_ff => reset,
38           rd_ff => RD,
39           wr_ff => WR,
40           equal_cont_wd_e_rd => saida_eq_cont_wr_rd,
41           equal_cont_wd_1_e_rd => saida_eq_cont_wr1_rd,
42           equal_cont_wd_e_rd_1 => saida_eq_cont_wr_rd1,
43           led_vazio_ff => l_vazio,
44           led_cheio_ff => l_cheio,
45           load_wr => saida_load_wr,
46           load_rd => saida_load_rd,
47           clr_ff => clr_dt,
48           saida_mv_fifo => saida_fifo);
49
50        Datapath: datapathFIFO port map(
51           clk_fifo_dp => CLK_fifo,
52           wr_data_dp => W_data,
53           ld_wr_dp => saida_load_wr,
54           ld_rd_dp => saida_load_rd,
55           clr_fifo_dp => clr_dt,
56           eq_comp_wr_rd => saida_eq_cont_wr_rd,
57           eq_comp_wr1_rd => saida_eq_cont_wr1_rd,
58           eq_comp_wr_rd1 => saida_eq_cont_wr_rd1,
59           rd_data_dp => leitura_banco_registradores);
60
61
62
63        R_data <= leitura_banco_registradores;
64        em <= l_vazio;
65        fu <= l_cheio;
66        Estados_maquina <= saida_fifo;
```

```
67
68    end ckt ;
```

```
1  library ieee;
2  use ieee.std_logic_1164 .all;
3
4  entity blocoDeControleFIFO is
5     port (clk_ff , rst_ff, rd_ff, wr_ff: in std_logic;
6           equal_cont_wd_e_rd, equal_cont_wd_1_e_rd, equal_cont_wd_e_rd_1: in std_logic;
7           led_vazio_ff, led_cheio_ff,load_wr,load_rd, clr_ff: out std_logic;
8           saida_mv_fifo: out std_logic_vector(2 downto 0));
9  end blocoDeControleFIFO;
10
11 architecture ckt of blocoDeControleFIFO is
12    type st is (start, p_leitura, escrita, p_escrita, leitura);
13    signal estado : st;
14    signal p_l,p_e: std_logic;
15    begin
16       process (clk_ff , rst_ff)
17       begin
18          if rst_ff = '1' then
19             estado <= start ;
20          elsif (clk_ff'event and clk_ff ='1') then
21             case estado is
22                when start =>
23                   estado <= p_leitura;
24                when p_leitura =>
25                   if (equal_cont_wd_e_rd='0') and (rd_ff ='1') and (wr_ff ='0') then
   estado <= leitura;
26                   elsif (wr_ff='1') then estado <= escrita;
27                   else estado <= p_leitura;
28                   end if;
29                when escrita =>
30                   if equal_cont_wd_1_e_rd='1' then estado <= p_escrita;
31                   elsif (wr_ff='1') and (rd_ff='0') then estado <= escrita;
32                   else estado <= p_escrita;
33                   end if;
34                when p_escrita =>
35                   if (equal_cont_wd_e_rd='0') and (wr_ff ='1') and (rd_ff='0') then estado
    <= escrita;
36                   elsif (rd_ff='1') then estado <= leitura;
37                   else estado <= p_escrita;
38                   end if;
39                when leitura =>
40                   if equal_cont_wd_e_rd_1='1' then estado <= p_leitura;
41                   elsif (wr_ff='0') and (rd_ff='1') then estado <= leitura;
42                   else estado <= p_leitura;
43                   end if;
44             end case ;
45          end if;
46       end process;
47       clr_ff <= '1' when estado = start else '0';
48       load_wr <= '1' when estado = escrita else '0';
49       load_rd <= '1' when estado = leitura else '0';
50       p_e <= '1' when estado = p_escrita else '0';
51       p_l <= '1' when estado = p_leitura else '0';
52
53       led_vazio_ff <= p_l and equal_cont_wd_e_rd;
54       led_cheio_ff <= p_e and equal_cont_wd_e_rd;
55
56       with estado select
57          saida_mv_fifo <= "000" when start ,
58          "001" when p_leitura,
59          "010" when escrita,
60          "011" when p_escrita,
61          "100" when leitura;
62 end ckt ;
```

```
1   library ieee ;
2   use ieee.std_logic_1164.all;
3
4   entity datapathFIFO is
5     port (wr_data_dp: in std_logic_vector(12 downto 0);
6           ld_wr_dp, ld_rd_dp, clr_fifo_dp, clk_fifo_dp: in std_logic;
7           eq_comp_wr_rd,eq_comp_wr1_rd,eq_comp_wr_rd1: out std_logic;
8           rd_data_dp: out std_logic_vector(12 downto 0));
9   end datapathFIFO;
10
11  architecture ckt of datapathFIFO is
12     component bancoDeRegistrador16X13 is
13       port (wr_data: in std_logic_vector(12 downto 0);
14             cont_wr,cont_rd: in std_logic_vector(3 downto 0);
15             clk_br, ld_wr, ld_rd, clr_reg: in std_logic;
16             rd_data: out std_logic_vector(12 downto 0));
17     end component;
18
19     component Comparador_4Bits is
20       port (eA,eB: in std_logic_vector(3 downto 0);
21             gt,lt,eq: in std_logic;
22             AeqB,AltB,AgtB: out std_logic);
23     end component;
24
25     component contador4Bits is
26       port ( clk_c4B,ld_c4B,clr_c4B: in std_logic;
27              out_c4B: out std_logic_vector(3 downto 0));
28     end component;
29
30     component SUM_4Bits is
31       port (A4_in,B4_in: in std_logic_vector(3 downto 0);
32             C4_in: in std_logic;
33             S4_out: out std_logic_vector(3 downto 0);
34             C4_out: out std_logic);
35     end component;
36
37     signal saida_cont_wr, saida_cont_rd: std_logic_vector(3 downto 0);
38     signal saida_sum_wr, saida_sum_rd: std_logic_vector(3 downto 0);
39     signal saida_bancoRegistradores: std_logic_vector(12 downto 0);
40     signal clr_reverso_reg: std_logic;
41     signal lixo: std_logic_vector(1 downto 0);
42     signal saida_eq_comparador_wd_com_rd, saida_ld_comparador_wd_com_rd,
    saida_gt_comparador_wd_com_rd: std_logic;
43     signal saida_eq_comparador_wd1_com_rd, saida_ld_comparador_wd1_com_rd,
    saida_gt_comparador_wd1_com_rd: std_logic;
44     signal saida_eq_comparador_wd_com_rd1, saida_ld_comparador_wd_com_rd1,
    saida_gt_comparador_wd_com_rd1: std_logic;
45
46     begin
47
48        clr_reverso_reg <= not clr_fifo_dp;
49
50        BancoDeRegistradores: bancoDeRegistrador16X13 port map(
51           wr_data => wr_data_dp,
52           cont_wr => saida_cont_wr,
53           cont_rd => saida_cont_rd,
54           clk_br => clk_fifo_dp,
55           ld_wr => ld_wr_dp,
56           ld_rd => ld_rd_dp,
57           clr_reg => clr_reverso_reg,
58           rd_data => saida_bancoRegistradores);
59
60        Contador_WR: contador4Bits port map(
61           clk_c4B => clk_fifo_dp,
62           ld_c4B => ld_wr_dp,
63           clr_c4B => clr_fifo_dp,
```

```
64            out_c4B => saida_cont_wr);
65
66         Contador_RD: contador4Bits port map(
67            clk_c4B => clk_fifo_dp,
68            ld_c4B => ld_rd_dp,
69            clr_c4B => clr_fifo_dp,
70            out_c4B => saida_cont_rd);
71
72         comparador_wr_com_rd: Comparador_4Bits port map(
73            eA => saida_cont_wr,
74            eB => saida_cont_rd,
75            gt => '0',
76            lt => '0',
77            eq => '1',
78            AeqB => saida_eq_comparador_wd_com_rd,
79            AltB => saida_ld_comparador_wd_com_rd,
80            AgtB => saida_gt_comparador_wd_com_rd);
81
82         comparador_wr1_com_rd: Comparador_4Bits port map(
83            eA => saida_sum_wr,
84            eB => saida_cont_rd,
85            gt => '0',
86            lt => '0',
87            eq => '1',
88            AeqB => saida_eq_comparador_wd1_com_rd,
89            AltB => saida_ld_comparador_wd1_com_rd,
90            AgtB => saida_gt_comparador_wd1_com_rd);
91
92         comparador_wr_com_rd1: Comparador_4Bits port map(
93            eA => saida_cont_wr,
94            eB => saida_sum_rd,
95            gt => '0',
96            lt => '0',
97            eq => '1',
98            AeqB => saida_eq_comparador_wd_com_rd1,
99            AltB => saida_ld_comparador_wd_com_rd1,
100           AgtB => saida_gt_comparador_wd_com_rd1);
101
102        somador_wr1: SUM_4Bits port map(
103           A4_in => saida_cont_wr,
104           B4_in => "0001",
105           C4_in => '0',
106           S4_out => saida_sum_wr,
107           C4_out => lixo(0));
108
109        somador_rd1: SUM_4Bits port map(
110           A4_in => saida_cont_rd,
111           B4_in => "0001",
112           C4_in => '0',
113           S4_out => saida_sum_rd,
114           C4_out => lixo(1));
115
116
117        rd_data_dp <= saida_bancoRegistradores;
118        eq_comp_wr_rd <= saida_eq_comparador_wd_com_rd;
119        eq_comp_wr1_rd <= saida_eq_comparador_wd1_com_rd;
120        eq_comp_wr_rd1 <= saida_eq_comparador_wd_com_rd1;
121
122  end ckt;
```

```
1   library ieee ;
2   use ieee.std_logic_1164.all;
3
4   entity bancoDeRegistrador16X13 is
5     port (wr_data: in std_logic_vector(12 downto 0);
6           cont_wr,cont_rd: in std_logic_vector(3 downto 0);
7           clk_br, ld_wr, ld_rd, clr_reg: in std_logic;
8           rd_data: out std_logic_vector(12 downto 0));
9   end bancoDeRegistrador16X13;
10
11  architecture ckt of bancoDeRegistrador16X13 is
12     component MUX16_1_13Bits is
13        port (I15_16, I14_16, I13_16, I12_16, I11_16, I10_16, I9_16, I8_16, I7_16:
    std_logic_vector(12 downto 0);
14              I6_16, I5_16, I4_16, I3_16, I2_16, I1_16, I0_16: std_logic_vector(12 downto 0);
15              S_16: in std_logic_vector(3 downto 0);
16              ld_mux_16: in std_logic;
17              d_16: out std_logic_vector(12 downto 0));
18     end component;
19
20     component decodificador1X16 is
21        port (ld_dec: in std_logic;
22              i_in: in std_logic_vector(3 downto 0);
23              d_out: out std_logic_vector(15 downto 0));
24     end component;
25
26     component reg13Bits is
27        port (clk,preSet,clr,load: in std_logic;
28              d: in std_logic_vector(12 downto 0);
29              q: out std_logic_vector(12 downto 0));
30     end component;
31
32
33     signal saida_reg_00, saida_reg_01, saida_reg_02, saida_reg_03, saida_reg_04,
    saida_reg_05, saida_reg_06, saida_reg_07: std_logic_vector(12 downto 0);
34     signal saida_reg_08, saida_reg_09, saida_reg_10, saida_reg_11, saida_reg_12,
    saida_reg_13, saida_reg_14, saida_reg_15, saida_mux: std_logic_vector(12 downto 0);
35     signal saida_decod: std_logic_vector(15 downto 0);
36
37     begin
38        Dec_wr: decodificador1X16 port map(
39           ld_dec => ld_wr,
40           i_in => cont_wr,
41           d_out => saida_decod);
42
43        Reg00: reg13Bits port map(
44           clk => clk_br,
45           preSet => '1',
46           clr => clr_reg,
47           load => saida_decod(0),
48           d => wr_data,
49           q => saida_reg_00);
50
51        Reg01: reg13Bits port map(
52           clk => clk_br,
53           preSet => '1',
54           clr => clr_reg,
55           load => saida_decod(1),
56           d => wr_data,
57           q => saida_reg_01);
58
59        Reg02: reg13Bits port map(
60           clk => clk_br,
61           preSet => '1',
62           clr => clr_reg,
63           load => saida_decod(2),
```

```
64            d => wr_data,
65            q => saida_reg_02);
66
67         Reg03: reg13Bits port map(
68            clk => clk_br,
69            preSet => '1',
70            clr => clr_reg,
71            load => saida_decod(3),
72            d => wr_data,
73            q => saida_reg_03);
74
75         Reg04: reg13Bits port map(
76            clk => clk_br,
77            preSet => '1',
78            clr => clr_reg,
79            load => saida_decod(4),
80            d => wr_data,
81            q => saida_reg_04);
82
83         Reg05: reg13Bits port map(
84            clk => clk_br,
85            preSet => '1',
86            clr => clr_reg,
87            load => saida_decod(5),
88            d => wr_data,
89            q => saida_reg_05);
90
91         Reg06: reg13Bits port map(
92            clk => clk_br,
93            preSet => '1',
94            clr => clr_reg,
95            load => saida_decod(6),
96            d => wr_data,
97            q => saida_reg_06);
98
99         Reg07: reg13Bits port map(
100           clk => clk_br,
101           preSet => '1',
102           clr => clr_reg,
103           load => saida_decod(7),
104           d => wr_data,
105           q => saida_reg_07);
106
107        Reg08: reg13Bits port map(
108           clk => clk_br,
109           preSet => '1',
110           clr => clr_reg,
111           load => saida_decod(8),
112           d => wr_data,
113           q => saida_reg_08);
114
115        Reg09: reg13Bits port map(
116           clk => clk_br,
117           preSet => '1',
118           clr => clr_reg,
119           load => saida_decod(9),
120           d => wr_data,
121           q => saida_reg_09);
122
123        Reg10: reg13Bits port map(
124           clk => clk_br,
125           preSet => '1',
126           clr => clr_reg,
127           load => saida_decod(10),
128           d => wr_data,
129           q => saida_reg_10);
```

```
130
131         Reg11: reg13Bits port map(
132            clk => clk_br,
133            preSet => '1',
134            clr => clr_reg,
135            load => saida_decod(11),
136            d => wr_data,
137            q => saida_reg_11);
138
139         Reg12: reg13Bits port map(
140            clk => clk_br,
141            preSet => '1',
142            clr => clr_reg,
143            load => saida_decod(12),
144            d => wr_data,
145            q => saida_reg_12);
146
147         Reg13: reg13Bits port map(
148            clk => clk_br,
149            preSet => '1',
150            clr => clr_reg,
151            load => saida_decod(13),
152            d => wr_data,
153            q => saida_reg_13);
154
155         Reg14: reg13Bits port map(
156            clk => clk_br,
157            preSet => '1',
158            clr => clr_reg,
159            load => saida_decod(14),
160            d => wr_data,
161            q => saida_reg_14);
162
163         Reg15: reg13Bits port map(
164            clk => clk_br,
165            preSet => '1',
166            clr => clr_reg,
167            load => saida_decod(15),
168            d => wr_data,
169            q => saida_reg_15);
170
171         MUX_rd: mux16_1_13Bits port map(
172            I15_16 => saida_reg_15,
173            I14_16 => saida_reg_14,
174            I13_16 => saida_reg_13,
175            I12_16 => saida_reg_12,
176            I11_16 => saida_reg_11,
177            I10_16 => saida_reg_10,
178            I9_16 => saida_reg_09,
179            I8_16 => saida_reg_08,
180            I7_16 => saida_reg_07,
181            I6_16 => saida_reg_06,
182            I5_16 => saida_reg_05,
183            I4_16 => saida_reg_04,
184            I3_16 => saida_reg_03,
185            I2_16 => saida_reg_02,
186            I1_16 => saida_reg_01,
187            I0_16 => saida_reg_00,
188            S_16 => cont_rd,
189            ld_mux_16 => ld_rd,
190            d_16 => saida_mux);
191
192         rd_data <= saida_mux;
193   end ckt;
```

```
1   library ieee ;
2   use ieee.std_logic_1164.all;
3
4   entity decodificador1X16 is
5     port (ld_dec: in std_logic;
6           i_in: in std_logic_vector(3 downto 0);
7           d_out: out std_logic_vector(15 downto 0));
8   end decodificador1X16;
9
10  architecture ckt of decodificador1X16 is
11    begin
12      d_out(0) <= (not (i_in(3))) and (not(i_in(2))) and (not (i_in(1))) and (not(i_in(0)))
    and ld_dec;
13      d_out(1) <= (not (i_in(3))) and (not(i_in(2))) and (not (i_in(1))) and     (i_in(0))
    and ld_dec;
14      d_out(2) <= (not (i_in(3))) and (not(i_in(2))) and      (i_in(1)) and  (not(i_in(0)))
    and ld_dec;
15      d_out(3) <= (not (i_in(3))) and (not(i_in(2))) and      (i_in(1)) and      (i_in(0))
    and ld_dec;
16      d_out(4) <= (not (i_in(3))) and     (i_in(2)) and  (not (i_in(1))) and (not(i_in(0)))
    and ld_dec;
17      d_out(5) <= (not (i_in(3))) and     (i_in(2)) and  (not (i_in(1))) and     (i_in(0))
    and ld_dec;
18      d_out(6) <= (not (i_in(3))) and     (i_in(2)) and       (i_in(1)) and  (not(i_in(0)))
    and ld_dec;
19      d_out(7) <= (not (i_in(3))) and     (i_in(2)) and       (i_in(1)) and      (i_in(0))
    and ld_dec;
20      d_out(8) <=      (i_in(3)) and (not(i_in(2))) and  (not (i_in(1))) and (not(i_in(0)))
    and ld_dec;
21      d_out(9) <=      (i_in(3)) and (not(i_in(2))) and  (not (i_in(1))) and     (i_in(0))
    and ld_dec;
22      d_out(10) <=     (i_in(3)) and (not(i_in(2))) and       (i_in(1)) and  (not(i_in(0)))
    and ld_dec;
23      d_out(11) <=     (i_in(3)) and (not(i_in(2))) and       (i_in(1)) and      (i_in(0))
    and ld_dec;
24      d_out(12) <=     (i_in(3)) and     (i_in(2)) and   (not (i_in(1))) and (not(i_in(0)))
    and ld_dec;
25      d_out(13) <=     (i_in(3)) and     (i_in(2)) and   (not (i_in(1))) and     (i_in(0))
    and ld_dec;
26      d_out(14) <=     (i_in(3)) and     (i_in(2)) and       (i_in(1)) and  (not(i_in(0)))
    and ld_dec;
27      d_out(15) <=     (i_in(3)) and     (i_in(2)) and       (i_in(1)) and      (i_in(0))
    and ld_dec;
28
29  end ckt;
```

```
1   library ieee ;
2   use ieee.std_logic_1164.all;
3
4   entity reg13Bits is
5      port (clk,preSet,clr,load: in std_logic;
6      d: in std_logic_vector(12 downto 0);
7      q : out std_logic_vector(12 downto 0));
8   end reg13Bits;
9   architecture ckt of reg13Bits is
10
11     signal qs: std_logic_vector(12 downto 0);
12
13     begin
14        process (clk ,preSet,clr)
15        begin
16           if preSet = '0' then qs <= "1111111111111";
17           elsif clr = '0' then qs <= "0000000000000";
18           elsif clk ='1' and clk ' event then
19              if load = '1' then
20                 qs <= d;
21              end if;
22           end if;
23        end process ;
24        q <= qs;
25  end ckt;
```

```
1   library ieee ;
2   use ieee.std_logic_1164.all;
3
4   entity MUX16_1_13Bits is
5     port (I15_16, I14_16, I13_16, I12_16, I11_16, I10_16, I9_16, I8_16, I7_16:
    std_logic_vector(12 downto 0);
6           I6_16, I5_16, I4_16, I3_16, I2_16, I1_16, I0_16: std_logic_vector(12 downto 0);
7           S_16: in std_logic_vector(3 downto 0);
8           ld_mux_16: in std_logic;
9           d_16: out std_logic_vector(12 downto 0));
10  end MUX16_1_13Bits;
11
12  Architecture ckt of MUX16_1_13Bits is
13    component MUX2_1_13Bits is
14      port (I_1,I_0: in std_logic_vector(12 downto 0);
15            S,ld_mux: in std_logic;
16            d: out std_logic_vector(12 downto 0));
17    end component;
18
19    signal saida_i0i1, saida_i2i3, saida_i4i5, saida_i6i7, saida_i8i9, saida_i10i11,
    saida_i12i13, saida_i14i15 : std_logic_vector(12 downto 0);
20    signal saida_S0102, saida_S0304, saida_S0506, saida_S0708 : std_logic_vector(12 downto 0
    );
21    signal saida_SS0102, saida_SS0304, saida_final : std_logic_vector(12 downto 0);
22
23    begin
24        muxI0I1: MUX2_1_13Bits port map(
25                 I_1 => I1_16,
26                 I_0 => I0_16,
27                 S => S_16(0),
28                 ld_mux => ld_mux_16,
29                 d => saida_i0i1);
30
31        muxI2I3: MUX2_1_13Bits port map(
32                 I_1 => I3_16,
33                 I_0 => I2_16,
34                 S => S_16(0),
35                 ld_mux => ld_mux_16,
36                 d => saida_i2i3);
37
38        muxI4I5: MUX2_1_13Bits port map(
39                 I_1 => I5_16,
40                 I_0 => I4_16,
41                 S => S_16(0),
42                 ld_mux => ld_mux_16,
43                 d => saida_i4i5);
44
45        muxI6I7: MUX2_1_13Bits port map(
46                 I_1 => I7_16,
47                 I_0 => I6_16,
48                 S => S_16(0),
49                 ld_mux => ld_mux_16,
50                 d => saida_i6i7);
51
52        muxI8I9: MUX2_1_13Bits port map(
53                 I_1 => I9_16,
54                 I_0 => I8_16,
55                 S => S_16(0),
56                 ld_mux => ld_mux_16,
57                 d => saida_i8i9);
58
59        muxI10I11: MUX2_1_13Bits port map(
60                 I_1 => I11_16,
61                 I_0 => I10_16,
62                 S => S_16(0),
63                 ld_mux => ld_mux_16,
```

```
64                d => saida_i10i11);
65
66          muxI12I13: MUX2_1_13Bits port map(
67                I_1 => I13_16,
68                I_0 => I12_16,
69                S => S_16(0),
70                ld_mux => ld_mux_16,
71                d => saida_i12i13);
72
73          muxI14I15: MUX2_1_13Bits port map(
74                I_1 => I15_16,
75                I_0 => I14_16,
76                S => S_16(0),
77                ld_mux => ld_mux_16,
78                d => saida_i14i15);
79
80          muxSaida0102: MUX2_1_13Bits port map(
81                I_1 => saida_i2i3,
82                I_0 => saida_i0i1,
83                S => S_16(1),
84                ld_mux => ld_mux_16,
85                d => saida_S0102);
86
87          muxSaida0304: MUX2_1_13Bits port map(
88                I_1 => saida_i6i7,
89                I_0 => saida_i4i5,
90                S => S_16(1),
91                ld_mux => ld_mux_16,
92                d => saida_S0304);
93
94          muxSaida0506: MUX2_1_13Bits port map(
95                I_1 => saida_i10i11,
96                I_0 => saida_i8i9,
97                S => S_16(1),
98                ld_mux => ld_mux_16,
99                d => saida_S0506);
100
101         muxSaida0708: MUX2_1_13Bits port map(
102               I_1 => saida_i14i15,
103               I_0 => saida_i12i13,
104               S => S_16(1),
105               ld_mux => ld_mux_16,
106               d => saida_S0708);
107
108         muxSSaida0102: MUX2_1_13Bits port map(
109               I_1 => saida_S0304,
110               I_0 => saida_S0102,
111               S => S_16(2),
112               ld_mux => ld_mux_16,
113               d => saida_SS0102);
114
115         muxSSaida0304: MUX2_1_13Bits port map(
116               I_1 => saida_S0708,
117               I_0 => saida_S0506,
118               S => S_16(2),
119               ld_mux => ld_mux_16,
120               d => saida_SS0304);
121         muxFinal: MUX2_1_13Bits port map(
122               I_1 => saida_SS0304,
123               I_0 => saida_SS0102,
124               S => S_16(3),
125               ld_mux => ld_mux_16,
126               d => saida_final);
127
128
129         d_16 <= saida_final;
```

```
130   end ckt;
```

```
1   library ieee ;
2   use ieee.std_logic_1164.all;
3
4   entity MUX2_1_13Bits is
5     port (I_1,I_0: std_logic_vector(12 downto 0);
6           S, ld_mux: in std_logic;
7           d: out std_logic_vector(12 downto 0));
8   end MUX2_1_13Bits;
9
10  Architecture ckt of MUX2_1_13Bits is
11
12  Begin
13    d(0) <= ((((not S) and I_0(0)) or (S and I_1(0)))) and ld_mux;
14    d(1) <= ((((not S) and I_0(1)) or (S and I_1(1)))) and ld_mux;
15    d(2) <= ((((not S) and I_0(2)) or (S and I_1(2)))) and ld_mux;
16    d(3) <= ((((not S) and I_0(3)) or (S and I_1(3)))) and ld_mux;
17    d(4) <= ((((not S) and I_0(4)) or (S and I_1(4)))) and ld_mux;
18    d(5) <= ((((not S) and I_0(5)) or (S and I_1(5)))) and ld_mux;
19    d(6) <= ((((not S) and I_0(6)) or (S and I_1(6)))) and ld_mux;
20    d(7) <= ((((not S) and I_0(7)) or (S and I_1(7)))) and ld_mux;
21    d(8) <= ((((not S) and I_0(8)) or (S and I_1(8)))) and ld_mux;
22    d(9) <= ((((not S) and I_0(9)) or (S and I_1(9)))) and ld_mux;
23    d(10) <= ((((not S) and I_0(10)) or (S and I_1(10)))) and ld_mux;
24    d(11) <= ((((not S) and I_0(11)) or (S and I_1(11)))) and ld_mux;
25    d(12) <= ((((not S) and I_0(12)) or (S and I_1(12)))) and ld_mux;
26
27  end ckt;
```

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity contador4Bits is
5    port ( clk_c4B,ld_c4B,clr_c4B: in std_logic;
6           out_c4B: out std_logic_vector(3 downto 0));
7  end contador4Bits ;
8  architecture ckt of contador4Bits is
9     component ffjk is
10       port (clk ,J ,K ,P ,C: in std_logic;
11             q: out std_logic);
12    end component;
13
14    signal q0_out,q1_out,q2_out,q3_out, clr_reverso: std_logic;
15    signal resp_and: std_logic_vector(2 downto 0);
16
17    begin
18
19       clr_reverso <= not clr_c4B;
20       Q0: ffjk port map(
21            clk => clk_c4B,
22            J => ld_c4B,
23            K => ld_c4B,
24            P => '1',
25            C => clr_reverso,
26            q => q0_out);
27
28       resp_and(0) <= q0_out and ld_c4B;
29
30       Q1: ffjk port map(
31            clk => clk_c4B,
32            J => resp_and(0),
33            K => resp_and(0),
34            P => '1',
35            C => clr_reverso,
36            q => q1_out);
37
38       resp_and(1) <= q1_out and resp_and(0);
39
40       Q2: ffjk port map(
41            clk => clk_c4B,
42            J => resp_and(1),
43            K => resp_and(1),
44            P => '1',
45            C => clr_reverso,
46            q => q2_out);
47
48       resp_and(2) <= q2_out and resp_and(1);
49
50       Q3: ffjk port map(
51            clk => clk_c4B,
52            J => resp_and(2),
53            K => resp_and(2),
54            P => '1',
55            C => clr_reverso,
56            q => q3_out);
57
58       out_c4B(0) <= q0_out;
59       out_c4B(1) <= q1_out;
60       out_c4B(2) <= q2_out;
61       out_c4B(3) <= q3_out;
62
63 end ckt ;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   ENTITY ffjk IS
5     port ( clk ,J ,K ,P ,C : in std_logic;
6              q : out std_logic );
7   END ffjk ;
8   ARCHITECTURE ckt OF ffjk IS
9   SIGNAL qS : std_logic;
10  BEGIN
11    PROCESS ( clk ,P ,C )
12    BEGIN
13      IF P = '0' THEN qS <= '1';
14      ELSIF C ='0' THEN qS <= '0';
15      ELSIF clk = '1' AND clk ' EVENT THEN
16        IF J = '1' AND K = '1' THEN qS <= NOT qS ;
17        ELSIF J = '1' AND K = '0' THEN qS <= '1';
18        ELSIF J = '0' AND K = '1' THEN qS <= '0';
19        END IF;
20      END IF;
21    END PROCESS ;
22  q <= qS ;
23  END ckt ;
```

```
1   library ieee ;
2   use ieee.std_logic_1164.all;
3
4   entity Comparador_4Bits is
5     port (eA,eB: in std_logic_vector(3 downto 0);
6           gt,lt,eq: in std_logic;
7           AeqB,AltB,AgtB: out std_logic);
8   end Comparador_4Bits;
9
10  architecture ckt of Comparador_4Bits is
11
12    component Comparador is
13      port (in_gt, in_eq, in_lt,a,b: in std_logic;
14            out_eq,out_lt,out_gt: out std_logic);
15     end component;
16
17    signal saida_gt,saida_lt,saida_eq:std_logic_vector(3 downto 0);
18    begin
19    Comp1:Comparador port map(
20          in_gt => gt,
21          in_lt => lt,
22          in_eq => eq,
23          a => eA(3),
24          b => eB(3),
25          out_eq => saida_eq(3),
26          out_gt => saida_gt(3),
27          out_lt => saida_lt(3));
28
29    Comp2:Comparador port map(
30          in_gt => saida_gt(3),
31          in_lt => saida_lt(3),
32          in_eq => saida_eq(3),
33          a => eA(2),
34          b => eB(2),
35          out_eq => saida_eq(2),
36          out_gt => saida_gt(2),
37          out_lt => saida_lt(2));
38
39    Comp3:Comparador port map(
40          in_gt => saida_gt(2),
41          in_lt => saida_lt(2),
42          in_eq => saida_eq(2),
43          a => eA(1),
44          b => eB(1),
45          out_eq => saida_eq(1),
46          out_gt => saida_gt(1),
47          out_lt => saida_lt(1));
48
49    Comp4:Comparador port map(
50          in_gt => saida_gt(1),
51          in_lt => saida_lt(1),
52          in_eq => saida_eq(1),
53          a => eA(0),
54          b => eB(0),
55          out_eq => saida_eq(0),
56          out_gt => saida_gt(0),
57          out_lt => saida_lt(0));
58
59    AeqB <= saida_eq(0);
60    AltB <= saida_lt(0);
61    AgtB <= saida_gt(0);
62  end ckt;
63
64
65
```

```
1   library ieee ;
2   use ieee.std_logic_1164.all;
3
4   entity Comparador is
5     port (in_gt, in_eq, in_lt,a,b: in std_logic;
6           out_eq,out_lt,out_gt: out std_logic);
7   end Comparador;
8
9   architecture ckt of Comparador is
10    begin
11      out_gt <= in_gt OR (in_eq AND a AND (NOT b));
12      out_lt <= in_lt OR (in_eq AND (NOT a) AND b);
13      out_eq <= in_eq AND (a XNOR b);
14  end ckt;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity SUM_4Bits is
5     port (A4_in,B4_in: in std_logic_vector(3 downto 0);
6           C4_in: in std_logic;
7           S4_out: out std_logic_vector(3 downto 0);
8           C4_out: out std_logic);
9   end SUM_4Bits;
10
11  architecture ckt of SUM_4Bits is
12    component SUM_2Bits is
13      port (A2_in,B2_in: in std_logic_vector(1 downto 0);
14            C2_in: in std_logic;
15            S2_out: out std_logic_vector(1 downto 0);
16            C2_out: out std_logic);
17    end component;
18
19    signal Sum_01_out : std_logic;
20
21    begin
22      SUM01: SUM_2Bits port map(
23            A2_in => A4_in(1 downto 0),
24            B2_in => B4_in(1 downto 0),
25            C2_in => C4_in,
26            S2_out => S4_out(1 downto 0),
27            C2_out => Sum_01_out);
28
29      SUM02: SUM_2Bits port map(
30            A2_in => A4_in(3 downto 2),
31            B2_in => B4_in(3 downto 2),
32            C2_in => Sum_01_out,
33            S2_out => S4_out(3 downto 2),
34            C2_out => C4_out);
35
36  end ckt;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity SUM_2Bits is
5     port (A2_in,B2_in: in std_logic_vector(1 downto 0);
6           C2_in: in std_logic;
7           S2_out: out std_logic_vector(1 downto 0);
8           C2_out: out std_logic);
9   end SUM_2Bits;
10
11  architecture ckt of SUM_2Bits is
12    component SUM_1Bit is
13      port (A_in,B_in,C_in: in std_logic;
14            S_out, C_out: out std_logic);
15    end component;
16
17    signal Sum_01_out : std_logic;
18
19    begin
20      SUM01: SUM_1Bit port map(
21            A_in => A2_in(0),
22            B_in => B2_in(0),
23            C_in => C2_in,
24            S_out => S2_out(0),
25            C_out => Sum_01_out);
26
27      SUM02: SUM_1Bit port map(
28            A_in => A2_in(1),
29            B_in => B2_in(1),
30            C_in => Sum_01_out,
31            S_out => S2_out(1),
32            C_out => C2_out);
33
34  end ckt;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity SUM_1Bit is
5     port (A_in,B_in,C_in: in std_logic;
6           S_out, C_out: out std_logic);
7   end SUM_1Bit;
8
9   Architecture ckt of SUM_1Bit is
10
11  Begin
12    S_out <= ((B_in and ((C_in nor A_in) or (C_in and A_in))) or ((not B_in) and (((not C_in
    ) and A_in) or (C_in and (not A_in)))));
13    C_out <= ((C_in and (A_in or B_in)) or (A_in and B_in));
14  end ckt;
```