```
1  ----------------------------------------------------------------------------------
2  -- Company:        --
3  -- Engineer:       Dipl.-Ing. Roland Nickel
4  --
5  -- Create Date:    19:29:24 12/29/2009
6  -- Design Name:    Roland
7  -- Module Name:    dcf1 - Behavioral
8  -- Project Name:   DCF Decoder + Anzeige
9  -- Target Devices: XC95144XL-10-TQ100
10 -- Tool versions:  0.01
11 -- Description:    DCF Empfänger Pollin
12 --                 CPLD Board Pollin
13 --                 LCD-Modul JM082A
14 -- Dependencies:
15 --
16 -- Revision:       0.01
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19 --
20 ----------------------------------------------------------------------------------
21 library IEEE;
22 use IEEE.STD_LOGIC_1164.ALL;
23 use IEEE.STD_LOGIC_ARITH.ALL;
24 use IEEE.STD_LOGIC_UNSIGNED.ALL;
25
26 ---- Uncomment the following library declaration if instantiating
27 ---- any Xilinx primitives in this code.
28 library UNISIM;
29 use UNISIM.VComponents.all;
30
31 entity dcf1 is
32     Port ( clk  : in  STD_LOGIC; -- Clock
33            data : in  STD_LOGIC; -- Data
34            led1 : out STD_LOGIC; -- LED's
35            led2 : out STD_LOGIC;
36            e    : out STD_LOGIC;
37            rw   : out STD_LOGIC;
38            rs   : out STD_LOGIC;
39            d0   : out STD_LOGIC;
40            d1   : out STD_LOGIC;
41            d2   : out STD_LOGIC;
42            d3   : out STD_LOGIC;
43            d4   : out STD_LOGIC;
44            d5   : out STD_LOGIC;
45            d6   : out STD_LOGIC;
46            d7   : out STD_LOGIC);
47 end dcf1;
48
49 architecture Behavioral of dcf1 is
50    signal cnt     : STD_LOGIC_VECTOR (13 downto 0);   -- Zähler
51    signal cnt_1s  : STD_LOGIC_VECTOR (10 downto 0);
52    SIGNAL clk_i1  : STD_LOGIC_VECTOR (7 downto 0);
53    SIGNAL clk_i2  : STD_LOGIC_VECTOR (7 downto 0);
54    SIGNAL clk_i3  : STD_LOGIC_VECTOR (4 downto 0);
55
56    signal clk_ms1 : STD_LOGIC;                        -- Clk Ltg.
57    signal clk_ms2 : STD_LOGIC;
58    signal data_1  : STD_LOGIC;
59    signal data_2  : STD_LOGIC;
60    signal data_3  : STD_LOGIC;
61    signal clk_se  : STD_LOGIC;
```

```
62          signal sync    : STD_LOGIC;
63          signal clk_lcd : STD_LOGIC;
64          signal clk_dp  : STD_LOGIC;
65
66          SIGNAL ser_dat : STD_LOGIC_VECTOR (58 downto 0);   -- Bus Ltg.
67          SIGNAL data_r1 : STD_LOGIC_VECTOR (6 downto 0);
68          SIGNAL data_r2 : STD_LOGIC_VECTOR (5 downto 0);
69          SIGNAL lcd_out : STD_LOGIC_VECTOR (10 downto 0);
70
71       begin
72          ---------------------------------------------------------------------------
73          -- Erzeugt clk_ms2 mit 1ms Taktzeit
74          -- Daten werden zwischengespeichert
75          ---------------------------------------------------------------------------
76          DCF_VT: PROCESS (clk, cnt)
77          BEGIN
78             if rising_edge(clk) then
79                if data = '0' then               -- Daten mit Takt synchronisieren
80                   data_1 <= '0';
81                else
82                   data_1 <= '1';
83                end if;
84
85                if clk_se = '1' or data = '1' then
86                   clk_dp <= '1';
87                else
88                   clk_dp <= '0';
89                end if;
90
91                clk_lcd <= cnt(9);            -- LCD Ausgabe Clock !!!!!
92
93                if cnt = 8000 then
94                   cnt <= (others => '0');
95                   clk_ms1 <= not clk_ms1;       -- ms Takt
96                else
97                   cnt <= cnt + 1;
98                end if;
99             end if;
100
101            if falling_edge(clk) then
102               clk_ms2 <= clk_ms1;              -- ms Takt
103               data_2 <= data_1;                -- Impulse
104            end if;
105         END PROCESS DCF_VT;
106
107         ---------------------------------------------------------------------------
108         -- Impulslänge zählen in ms Schritten
109         ---------------------------------------------------------------------------
110         DCF_CT1: PROCESS (clk_ms2, clk_i1)
111         BEGIN
112            if rising_edge(clk_ms2) then        -- akt. Counter übergeben
113               data_3 <= data_2;                -- Impuls
114               clk_i2 <= clk_i1;                -- Impulslänge
115
116               if data_2 = '0' then             -- Daten = 0 ??
117                  clk_i1 <= (others => '0');    -- ja -> Counter = 0
118               else
119                  if clk_i1 < 250 then          -- nein Counter + 1 (max 250)
120                     clk_i1 <= clk_i1 + 1;
121                  end if;
122               end if;
```

```
123            end if;
124         END PROCESS DCF_CT1;
125
126         -------------------------------------------------------------------------
127         -- 1 Sekunde --> auszählen in ms Schritten
128         -------------------------------------------------------------------------
129         DCF_CT2: PROCESS (clk_ms2, cnt_1s)
130         BEGIN
131            if rising_edge(clk_ms2) then
132               if data_2 = '0' then             -- Sekundenimpuls = 0 ??
133                  cnt_1s <= cnt_1s + 1;         -- ms zählen
134               else
135                  cnt_1s <= (others => '0');    -- Zähler für Impulse -> RESET
136               end if;
137            end if;
138
139            if falling_edge(clk_ms2) then
140               if cnt_1s = 1250 then            -- Auswertung der Dauer
141                  clk_se <= '1';                -- Ladeimpuls für Register
142               else
143                  clk_se <= '0';
144               end if;
145            end if;
146         END PROCESS DCF_CT2;
147
148         -------------------------------------------------------------------------
149         -- LCD Ausgabe
150         -------------------------------------------------------------------------
151         DCF_LCD: PROCESS (clk_lcd)
152         BEGIN
153            if rising_edge(clk_lcd) then
154               if sync = '1' then
155                  if clk_i3 = 31 then
156                     clk_i3 <= "01001";
157                  else
158                     clk_i3 <= clk_i3 + 1;
159                  end if;
160                  ---------------------------------------------------------------
161                  -- Initialisierung
162                  ---------------------------------------------------------------
163                  case clk_i3 is
164                  WHEN "00001" =>
165                     --          SW Function set
166                     lcd_out <= "00100110100";
167                  WHEN "00011" =>
168                     --          SW Display ON/OFF Control
169                     lcd_out <= "00100001100";
170                  WHEN "00101" =>
171                     --          SW Clear Display
172                     lcd_out <= "00100000001";
173                  WHEN "00111" =>
174                     --          SW Entry Mode Set
175                     lcd_out <= "00100000110";
176                  ---------------------------------------------------------------
177                  -- Daten aufs LCD-Modul ausgeben
178                  ---------------------------------------------------------------
179                  WHEN "01001" =>
180                     --          SW Set DDRAM Address
181                     lcd_out <= "00110000001";
182                  WHEN "01011" =>
183                     lcd_out(0)  <= data_r2(4);    -- Stunden Hi
```

```
184                     lcd_out(1)  <= data_r2(5);
185                     lcd_out(2)  <= '0';
186                     lcd_out(3)  <= '0';
187                     lcd_out(4)  <= '1';
188                     if data_r2(4) = '0' and data_r2(5) = '0' then
189                        lcd_out(5) <= '0';
190                     else
191                        lcd_out(5) <= '1';
192                     end if;
193                     lcd_out(6)  <= '0';
194                     lcd_out(7)  <= '0';
195                     lcd_out(8)  <= '1'; -- E
196                     lcd_out(9)  <= '0'; -- W
197                     lcd_out(10) <= '1'; -- D
198                  WHEN "01101" =>
199                     lcd_out(0)  <= data_r2(0);    -- Stunden Lo
200                     lcd_out(1)  <= data_r2(1);
201                     lcd_out(2)  <= data_r2(2);
202                     lcd_out(3)  <= data_r2(3);
203                     lcd_out(4)  <= '1';
204                     lcd_out(5)  <= '1';
205                     lcd_out(6)  <= '0';
206                     lcd_out(7)  <= '0';
207                     lcd_out(8)  <= '1'; -- E
208                     lcd_out(9)  <= '0'; -- W
209                     lcd_out(10) <= '1'; -- D
210                  WHEN "01111" =>                  -- Trennung
211                     if clk_dp = '1' then
212                        lcd_out(1)  <= '1';        -- Doppelpunkt
213                        lcd_out(3)  <= '1';
214                        lcd_out(4)  <= '1';
215                     else
216                        lcd_out(1)  <= '0';        -- Leerzeichen
217                        lcd_out(3)  <= '0';
218                        lcd_out(4)  <= '0';
219                     end if;
220                     lcd_out(0)  <= '0';
221                     lcd_out(2)  <= '0';
222                     lcd_out(5)  <= '1';
223                     lcd_out(6)  <= '0';
224                     lcd_out(7)  <= '0';
225                     lcd_out(8)  <= '1'; -- E
226                     lcd_out(9)  <= '0'; -- W
227                     lcd_out(10) <= '1'; -- D
228                  WHEN "10001" =>
229                     lcd_out(0)  <= data_r1(4);    -- Minuten Hi
230                     lcd_out(1)  <= data_r1(5);
231                     lcd_out(2)  <= data_r1(6);
232                     lcd_out(3)  <= '0';
233                     lcd_out(4)  <= '1';
234                     lcd_out(5)  <= '1';
235                     lcd_out(6)  <= '0';
236                     lcd_out(7)  <= '0';
237                     lcd_out(8)  <= '1'; -- E
238                     lcd_out(9)  <= '0'; -- W
239                     lcd_out(10) <= '1'; -- D
240                  WHEN "10011" =>
241                     lcd_out(0)  <= data_r1(0);    -- Minuten Lo
242                     lcd_out(1)  <= data_r1(1);
243                     lcd_out(2)  <= data_r1(2);
244                     lcd_out(3)  <= data_r1(3);
```

```
245                     lcd_out(4)  <= '1';
246                     lcd_out(5)  <= '1';
247                     lcd_out(6)  <= '0';
248                     lcd_out(7)  <= '0';
249                     lcd_out(8)  <= '1'; -- E
250                     lcd_out(9)  <= '0'; -- W
251                     lcd_out(10) <= '1'; -- D
252                  ------------------------------------------------------------------
253                  -- Impuls zur Übernahme erzeugen
254                  ------------------------------------------------------------------
255                  WHEN OTHERS =>
256                     if clk_i3 < 21 then
257                        lcd_out(8)  <= '0';        -- Impuls
258                     else
259                        lcd_out(8)  <= '1';
260                     end if;
261                  end case;
262                  ------------------------------------------------------------------
263               end if;
264            end if;
265         END PROCESS DCF_LCD;
266
267         ------------------------------------------------------------------------
268         -- Daten / Impulslänge auswerten
269         ------------------------------------------------------------------------
270         DCF_AW: PROCESS (data_3)
271         BEGIN
272            if falling_edge(data_3) then
273               for i in 0 to 57 loop               -- Datenbit 1x schieben
274                  ser_dat(i) <= ser_dat(i+1);
275               end loop;
276
277               if clk_i2 > 130 then                -- Counter auswerten
278                  ser_dat(58) <= '1';              -- binäre 1 empfangen
279               else
280                  ser_dat(58) <= '0';              -- binäre 0 empfangen
281               end if;
282            end if;
283         END PROCESS DCF_AW;
284
285         ------------------------------------------------------------------------
286         -- Datenübernahme bei fehlender Sekunde
287         ------------------------------------------------------------------------
288         DCF_1M: PROCESS (clk_se)
289         BEGIN
290            if rising_edge(clk_se) then             -- fehlender Sekundenimpuls
291               if ser_dat(0) = '0' and ser_dat(20) = '1' then
292                  data_r1(0) <= ser_dat(21);       -- BCD Minuten (DCF)
293                  data_r1(1) <= ser_dat(22);
294                  data_r1(2) <= ser_dat(23);
295                  data_r1(3) <= ser_dat(24);
296                  data_r1(4) <= ser_dat(25);
297                  data_r1(5) <= ser_dat(26);
298                  data_r1(6) <= ser_dat(27);
299
300                  data_r2(0) <= ser_dat(29);       -- BCD Stunden (DCF)
301                  data_r2(1) <= ser_dat(30);
302                  data_r2(2) <= ser_dat(31);
303                  data_r2(3) <= ser_dat(32);
304                  data_r2(4) <= ser_dat(33);
305                  data_r2(5) <= ser_dat(34);
```

```
306
307                  sync <= '1';
308               else
309                  sync <= '0';
310               end if;
311            end if;
312         END PROCESS DCF_1M;
313
314         ----------------------------------------------------------------------------
315         -- LED Ausgänge
316         ----------------------------------------------------------------------------
317         led1 <= not clk_dp;                          -- Sekunden Impuls
318         led2 <= not sync;                            -- Daten gefunden
319
320         ----------------------------------------------------------------------------
321         -- LCD Ausgänge
322         ----------------------------------------------------------------------------
323         d0 <= lcd_out(0);
324         d1 <= lcd_out(1);
325         d2 <= lcd_out(2);
326         d3 <= lcd_out(3);
327         d4 <= lcd_out(4);
328         d5 <= lcd_out(5);
329         d6 <= lcd_out(6);
330         d7 <= lcd_out(7);
331         e  <= lcd_out(8);
332         rw <= lcd_out(9);
333         rs <= lcd_out(10);
334
335      end Behavioral;
336
337
```