```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:
6  -- Design Name:
7  -- Module Name:
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity Blinker is
31
32    Port (
33       clock: in std_logic;
34       reset: in std_logic;
35
36       push: in std_logic;
37
38
39       LED_Rot: out std_logic;
40
41
42       ------------------------------------------------------------
43       --       Hier die Externen Variablen zur beobachtung    -----
44       ------------------------------------------------------------
45
46       Sim_BCD_int: out std_logic_vector (1 downto 0);
47
48       Sim_Flanke_Push: out std_logic;
49       Sim_Takt_1_HZ: out std_logic;
50       Sim_Flanke_Takt_1HZ: out std_logic;
51
52       Sim_Start: out std_logic;
53
54       Sim_Counter_Takt: out std_logic_vector (25 downto 0)--; --Takt-Counter
55
56    );
57
58 end Blinker;
59
60 architecture Behavioral of Blinker is
61
```

```
62        -- Deklaration der Automatenzustände und internen Signale
63
64        type Zustaende is (a,b,c);
65
66
67        signal Push_Zustand, Push_Folgezustand: Zustaende;
68        signal Takt_1HZ_Zustand, Takt_1HZ_Folgezustand: Zustaende;
69
70
71        signal BCD_int: std_logic_vector (1 downto 0):="00"; --interne Zähler
72
73        signal Start:std_logic;--Der Countdown wurde gestartet
74        signal Takt_1_HZ:std_logic;
75        signal Counter_Takt: std_logic_vector (25 downto 0):="00000000000000000000000000";
          --Takt-Counter
76
77
78        signal Flanke_Push: std_logic;
79        signal Flanke_Takt_1HZ: std_logic;
80
81        begin
82
83
84
85
86        Zustandsmaschine: Process (clock, reset)
87           begin
88              if reset = '1'
89                 then  Push_Zustand <= a;
90                       Takt_1HZ_Zustand <=a;
91
92              elsif clock = '1' and clock'event
93                 then  Push_Zustand <= Push_Folgezustand;
94                       Takt_1HZ_Zustand <= Takt_1HZ_Folgezustand;
95              end if;
96           end process Zustandsmaschine;
97
98        Flankenerkennung_Push: Process (Push, Push_Zustand)
99              begin
100                case Push_Zustand is
101                   when a => if Push='1' then Push_Folgezustand <=a;
102                             else             Push_Folgezustand <=b;
103                             end if;
104                             Flanke_Push <= '0';
105                   when b => if Push='0' then Push_Folgezustand <=b;
106                             else             Push_Folgezustand <=c;
107                             end if;
108                             Flanke_Push <= '0';
109                   when c => if Push='1' then Push_Folgezustand <=a;
110                             else             Push_Folgezustand <=b;
111                             end if;
112                             Flanke_Push <= '1';
113                end case;
114             end process Flankenerkennung_Push;
115
116
117       Flankenerkennung_Takt_1HZ: Process (Takt_1_HZ, Takt_1HZ_Zustand)
118             begin
119                case Takt_1HZ_Zustand is
120                   when a => if Takt_1_HZ='1' then Takt_1HZ_Folgezustand <=a;
121                             else             Takt_1HZ_Folgezustand <=b;
```

```
122                              end if;
123                              Flanke_Takt_1HZ <= '0';
124                     when b => if Takt_1_HZ='0' then Takt_1HZ_Folgezustand <=b;
125                              else                Takt_1HZ_Folgezustand <=c;
126                              end if;
127                              Flanke_Takt_1HZ <= '0';
128                     when c => if Takt_1_HZ='1' then Takt_1HZ_Folgezustand <=a;
129                              else                Takt_1HZ_Folgezustand <=b;
130                              end if;
131                              Flanke_Takt_1HZ <= '1';
132                end case;
133             end process Flankenerkennung_Takt_1HZ;
134
135
136
137          Counter_Start : process (reset, clock, Flanke_Push)
138             begin
139                if reset ='1' then Start <= '0';
140                elsif clock='1' and clock'event then
141                   if Flanke_Push ='1' then
142                      Start <='1';
143                   elsif BCD_int = 0 then
144                      Start <='0';
145                   end if;
146                end if;
147             end process Counter_Start;
148
149
150
151          BCD_Counter : process (reset, clock, Start, Flanke_Push, BCD_int, Flanke_Takt_1HZ)
152             begin
153                if reset ='1' then BCD_int <= "11";
154                elsif clock='1' and clock'event then
155                   if (Start ='1' and Flanke_Takt_1HZ='1') then
156                      if BCD_int = 0 then BCD_int <= BCD_int;--Dann sind wir wohl bei Null
157                      else BCD_int <= BCD_int - 1;
158                      end if;
159                   elsif (Start='0')then
160                      BCD_int<="11";
161                   else BCD_int <= BCD_int;
162                   end if;
163                end if;
164             end process BCD_Counter;
165
166          -------------------------Taktteiler-------------------------------------
167          --   damit wir auf 0,1 Sekunden kommen müssen wir den Takt durch 50.000.000 teilen
168          --   wir brauchen log2 (50.000.000) = 26 Stellen (aufgerundet)
169          ------------------------------------------------------------------------
170          Takt_Teiler : process (reset, clock, Takt_1_HZ)
171             begin
172                if reset ='1' then
173                   Counter_Takt <="00000000000000000000000000";
174
175                elsif clock='1' and clock'event then
176                      if Takt_1_HZ = '1' then
177                         Counter_Takt <= Counter_Takt + 1;--einfach weiterzählen
178                      else
179                         Counter_Takt <= Counter_Takt - 1;--einfach weiterzählen
180                      end if;
181                end if;
182             end process Takt_Teiler;
```

```
183
184
185
186         -------------------------Ende Taktteiler-------------------------------
187
188
189
190         -- Hier jetzt die Asyncronen Anweisungen die den Übertrag und die Ausgänge aktualisieren
191
192
193         Takt_1_HZ <=  '0' when Counter_Takt = 50000000 else
194                       '1' when Counter_Takt = 0 else
195                       Takt_1_HZ;
196
197
198
199
200
201         LED_Rot <= '1' when (Takt_1_HZ ='1' and Start = '0') else
202                    '0';
203
204         ---------------------------------------------------------------------
205         ---            Ausgabe der internen Signale auf Ausgängen           ---
206         ---------------------------------------------------------------------
207
208
209               Sim_BCD_int <= BCD_int;
210
211               Sim_Flanke_Push <= Flanke_Push;
212               Sim_Counter_Takt <= Counter_Takt; --Takt-Counter
213               Sim_Flanke_Takt_1HZ <= Flanke_Takt_1HZ;
214               sim_takt_1_hz <= takt_1_hz;
215               sim_start <= start;
216
217
218
219         end Behavioral;
220
221
```