```
1  --*************************************************************************************
2  -- Product: EZ1KUSB
3  -- Filename: sample1.vhd
4  -- Date:     04-27-2003
5
6  -- EasyFPGA
7  -- 905 Shell Blvd.,#202N
8  -- Foster City, CA 94404
9  -- USA
10 -- support@easyfpga.com
11 -- If this code works then it was written by Peter Bolech. If not, I don't know who wrote it.
12 --*************************************************************************************
13 library ieee;
14 use ieee.std_logic_1164.all;
15 use ieee.std_logic_unsigned.all;
16
17 entity sample1 is port (
18
19   -- Clock signals and reset --
20   LCLK:                in std_logic;                       -- clock 10 MHz, on-board clock
21   nRESET:              in std_logic;                       -- FPGA reset, on-board reset
22
23   -- FT245BM USB FIFO interface --
24   USB_D:               inout std_logic_vector(7 downto 0);  -- FT245BM data bus
25   nUSB_RD:             inout std_logic;                    -- FT245BM read enable
26   USB_WR:              inout std_logic;                    -- FT245BM write enable
27   nUSB_TXE:            in std_logic;                       -- FT245BM transmit FIFO empty flag
28   nUSB_RXF:            in std_logic;                       -- FT245BM receive FIFO full flag
29
30   -- LEDs
31   LED:                 out std_logic_vector(7 downto 0);    -- on-board LEDs
32
33   -- I/O headers
34   IO:                  inout std_logic_vector(31 downto 0); -- I/O pins
35
36   -- EXP header
37   EXT_CLOCK:           in std_logic;                       -- external clock
38   EXP_IN0:             in std_logic;                       -- external dedicated input pin
39   EXP_IN1:             in std_logic;                       -- external dedicated input pin
40   EXP_IN2:             in std_logic;                       -- external dedicated input pin
41   EXP_IO:              inout std_logic_vector(24 downto 0)  -- I/O pins
42
43    );
44   end sample1;
45
46
47
```

```
48 architecture sample1_arch of sample1 is
49
50 signal COUNTER:          std_logic_vector(2 downto 0);
51 signal COUNT_ENAB:       std_logic;
52 signal PRESCALER:        std_logic_vector(21 downto 0);
53 signal VALUE:            std_logic_vector(21 downto 0);
54 signal REG:              std_logic_vector(7 downto 0);
55

56 begin
57
58 IO <= (others => 'Z');
59 EXP_IO <= (others => 'Z');
60
61 --*************************************************************************************************
62 --
63 --
64 USB_RD_PR: process(LCLK)
65 begin
66
67 if(LCLK'event and LCLK = '1') then
68  if(nUSB_RD = '0') then
69   nUSB_RD <= '1';
70  elsif(nUSB_RXF = '0') then
71   nUSB_RD <= '0';
72  else
73   nUSB_RD <= '1';
74  end if;
75 end if;
76
77 end process USB_RD_PR;
78
79 --*************************************************************************************************
80 --
81 --
82 USB_WR_PR: process(LCLK)
83 begin
84
85 if(LCLK'event and LCLK = '1') then
86   USB_WR <= '0';
87 end if;
88
89 end process USB_WR_PR;
90
91
92
93
```

```
--*****************************************************************************************
-- access to the register
--
USB_REG_WRITE_PR: process(LCLK)
begin

 if(LCLK'event and LCLK = '1') then
  if(nUSB_RD = '0') then
   REG <= USB_D;
  end if;
 end if;

 USB_D <= (others => 'Z');

end process USB_REG_WRITE_PR;

--*****************************************************************************************
-- Walking-one
--
LED_PR: process(LCLK)
begin

 if(LCLK'event and LCLK = '1') then
  if(nRESET = '0') then
   VALUE <= "00011110100001001000000";   -- 50 ms
  elsif(REG = "00000000") then
   VALUE <= "00011110100001001000000";   -- 50 ms
  elsif(REG = "00000001") then
   VALUE <= "00111101000010010000000";   -- 100 ms
  elsif(REG = "00000010") then
   VALUE <= "01111010000100100000000";   -- 200 ms
  end if;
 end if;

 -- prescaler counter
 if(LCLK'event and LCLK = '1') then
  if(nRESET = '0') then
   PRESCALER <= (others => '0');
  elsif(PRESCALER = VALUE) then
   PRESCALER <= (others => '0');
  else
   PRESCALER <= PRESCALER + '1';
  end if;
 end if;


```

```
141 -- clock enable
142  if(LCLK'event and LCLK = '1') then
143   if(PRESCALER = VALUE) then
144    COUNT_ENAB <= '1';
145   else
146    COUNT_ENAB <= '0';
147   end if;
148  end if;
149
150 -- 3 bit binary counter: 0, 1, 2, 3, 4, 5, 6, 7
151  if(LCLK'event and LCLK = '1') then
152   if(nRESET = '0') then
153    COUNTER <= (others => '0');
154   elsif(COUNT_ENAB = '1') then
155    COUNTER <= COUNTER + '1';
156   end if;
157  end if;
158
159 -- mux
160  if(LCLK'event and LCLK = '1') then
161   case COUNTER is
162    when "000" =>
163     LED <= "00000001";  -- LED 1
164    when "001" =>
165     LED <= "00000010";  -- LED 2
166    when "010" =>
167     LED <= "00000100";  -- LED 3
168    when "011" =>
169     LED <= "00001000";  -- LED 4
170    when "100" =>
171     LED <= "10000000";  -- LED 8
172    when "101" =>
173     LED <= "01000000";  -- LED 7
174    when "110" =>
175     LED <= "00100000";  -- LED 6
176    when "111" =>
177     LED <= "00010000";  -- LED 5
178    when others =>
179     LED <= "00000000";  -- remove latch !!!
180   end case;
181  end if;
182
183 end process LED_PR;
184
185 end sample1_arch;
186
187
```