```
--*************************************************************************************
-- Product: EZ1KUSB
-- Filename: sample2.vhd
-- Date:     08-01-2003

-- EasyFPGA
-- 905 Shell Blvd.,#202N
-- Foster City, CA 94404
-- USA
-- support@easyfpga.com
-- If this code works then it was written by Peter Bolech. If not, I don't know who wrote it.
--*************************************************************************************
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity sample2 is port (

  -- Clock signals and reset --
  LCLK:                 in std_logic;                       -- clock 10 MHz, on-board clock
  nRESET:               in std_logic;                       -- FPGA reset, on-board reset

  -- FT245BM USB FIFO interface --
  USB_D:                inout std_logic_vector(7 downto 0);  -- FT245BM data bus
  nUSB_RD:              inout std_logic;                     -- FT245BM read enable
  USB_WR:               inout std_logic;                     -- FT245BM write enable
  nUSB_TXE:             in std_logic;                        -- FT245BM transmit FIFO empty flag
  nUSB_RXF:             in std_logic;                        -- FT245BM receive FIFO full flag

  -- LEDs
  LED:                  out std_logic_vector(7 downto 0);    -- on-board LEDs

  -- I/O headers
  IO:                   inout std_logic_vector(31 downto 0); -- I/O pins

  -- EXP header
  EXT_CLOCK:            in std_logic;                        -- external clock
  EXP_IN0:              in std_logic;                        -- external dedicated input pin
  EXP_IN1:              in std_logic;                        -- external dedicated input pin
  EXP_IN2:              in std_logic;                        -- external dedicated input pin
  EXP_IO:               inout std_logic_vector(24 downto 0)  -- I/O pins

   );
  end sample2;
```

```
52
53 architecture sample2_arch of sample2 is
54
55 type STATE_MACHINE_TYPE is (wait_for_rxf_low_0, usb_read_0, wait_one_cycle_0,
56                             wait_for_rxf_low_1, usb_read_1, wait_one_cycle_1,
57                             wait_for_rxf_low_2, usb_read_2, wait_one_cycle_2,
58                             wait_for_rxf_low_3, usb_read_3, wait_one_cycle_3,
59                             local_lad, local_data, wait_for_txe_low, usb_write,
60                             wait_one_cycle_4);
61
62 signal STATE_MACHINE:    STATE_MACHINE_TYPE;
63 signal nUSB_TXE_LCH:     std_logic;
64 signal nUSB_RXF_LCH:     std_logic;
65 signal USB_D_LCH0:       std_logic_vector(7 downto 0);
66 signal USB_D_LCH1:       std_logic_vector(7 downto 0);
67 signal USB_D_LCH2:       std_logic_vector(7 downto 0);
68 signal USB_D_LCH3:       std_logic_vector(7 downto 0);
69
70 signal LAD_IN:           std_logic_vector(7 downto 0);
71 signal LAD_OUT:          std_logic_vector(7 downto 0);
72 signal LAD_INT:          std_logic_vector(7 downto 0);
73
74 signal nCSINT:           std_logic;
75 signal nLADS:            std_logic;
76 signal nLRW:             std_logic;
77 signal WAIT_COUNTER:     std_logic_vector(2 downto 0);
78 signal PROC_ADD:         std_logic_vector(7 downto 0);
79 signal nLRDY:            std_logic;
80
81 signal FPGA_REV:         std_logic_vector(7 downto 0);
82 signal LED_REG:          std_logic_vector(7 downto 0);
83
84 constant FPGA_REVISION:  std_logic_vector(7 downto 0) := "00000001";
85
86 constant READ_COMMAND:   std_logic_vector(7 downto 0) := "10101010";
87 constant CS_INT_ADD:     std_logic_vector(7 downto 0) := "00000000";
88
89 constant FPGA_REV_ADD:   std_logic_vector(7 downto 0) := "00000000";
90 constant LED_REG_ADD:    std_logic_vector(7 downto 0) := "00000001";
91
92
93 begin
94
95 IO <= (others => 'Z');
96 EXP_IO <= (others => 'Z');
97
98
99
100
101
102
```

```
103
104 --*****************************************************************************************************
105 -- Byte 0: command (read/write)
106 -- Byte 1: chip select (CS_INT = "00000000")
107 -- Byte 2: address
108 -- Byte 3: if write - data, if read - don't care
109
110 STATE_MACHINE_PR: process (LCLK, nRESET)
111 begin
112
113 if(LCLK'event and LCLK = '1') then
114  if(nRESET = '0') then
115   STATE_MACHINE <= wait_for_rxf_low_0;
116  else
117  case STATE_MACHINE is
118
119 -- USB read byte 0
120    when wait_for_rxf_low_0 =>
121     if(nUSB_RXF_LCH = '0') then
122      STATE_MACHINE <= usb_read_0;
123     end if;
124
125    when usb_read_0 =>
126     STATE_MACHINE <= wait_one_cycle_0;
127
128    when wait_one_cycle_0 =>
129     STATE_MACHINE <= wait_for_rxf_low_1;
130
131 -- USB read byte 1
132    when wait_for_rxf_low_1 =>
133     if(nUSB_RXF_LCH = '0') then
134      STATE_MACHINE <= usb_read_1;
135     end if;
136
137    when usb_read_1 =>
138     STATE_MACHINE <= wait_one_cycle_1;
139
140    when wait_one_cycle_1 =>
141     STATE_MACHINE <= wait_for_rxf_low_2;
142
143 -- USB read byte 2
144    when wait_for_rxf_low_2 =>
145     if(nUSB_RXF_LCH = '0') then
146      STATE_MACHINE <= usb_read_2;
147     end if;
148
149    when usb_read_2 =>
150     STATE_MACHINE <= wait_one_cycle_2;
151
152    when wait_one_cycle_2 =>
153     STATE_MACHINE <= wait_for_rxf_low_3;
```

```
154
155 -- USB read byte 3
156    when wait_for_rxf_low_3 =>
157     if(nUSB_RXF_LCH = '0') then
158      STATE_MACHINE <= usb_read_3;
159     end if;
160
161    when usb_read_3 =>
162     STATE_MACHINE <= wait_one_cycle_3;
163
164    when wait_one_cycle_3 =>
165     STATE_MACHINE <= local_lad;
166
167 -- local bus access
168    when local_lad =>
169     STATE_MACHINE <= local_data;
170
171    when local_data =>
172     if(nLRDY = '0') then
173      if(nLRW = '0') then                       -- if local bus READ goto USB write
174       STATE_MACHINE <= wait_for_txe_low;
175      else
176       STATE_MACHINE <= wait_for_rxf_low_0;    -- if loacal bus WRITE goto "idle"
177      end if;
178     end if;
179
180 -- USB write
181    when wait_for_txe_low =>
182     if(nUSB_TXE_LCH = '0') then
183      STATE_MACHINE <= usb_write;
184     end if;
185
186    when usb_write =>
187     STATE_MACHINE <= wait_one_cycle_4;
188
189    when wait_one_cycle_4 =>
190     STATE_MACHINE <= wait_for_rxf_low_0;     -- data from local bus written into the ftdi chip, goto "idle"
191
192    when others =>
193     STATE_MACHINE <= wait_for_rxf_low_0;
194
195    end case;
196
197   end if;
198  end if;
199
200 end process STATE_MACHINE_PR;
201
202
203
204
```

```
205 --*************************************************************************************************************
206 -- nUSB_RD
207 USB_RD_PR: process(LCLK)
208 begin
209
210 if(LCLK'event and LCLK = '1') then
211  if(STATE_MACHINE = wait_for_rxf_low_0 or STATE_MACHINE = wait_for_rxf_low_1 or
212     STATE_MACHINE = wait_for_rxf_low_2 or STATE_MACHINE = wait_for_rxf_low_3) then
213      if(nUSB_RXF_LCH = '0') then
214       nUSB_RD <= '0';
215      else
216       nUSB_RD <= '1';
217      end if;
218   else
219    nUSB_RD <= '1';
220   end if;
221 end if;
222
223 end process USB_RD_PR;
224
225 --*************************************************************************************************************
226 -- USB_WR
227 USB_WR_PR: process(LCLK)
228 begin
229
230 if(LCLK'event and LCLK = '1') then
231  if(STATE_MACHINE = wait_for_txe_low) then
232   if(nUSB_TXE_LCH = '0') then
233    USB_WR <= '1';
234   else
235    USB_WR <= '0';
236   end if;
237  else
238   USB_WR <= '0';
239  end if;
240 end if;
241
242 end process USB_WR_PR;
243
244 --*************************************************************************************************************
245 -- sync. USB flags with local clock
246 USB_FLAGS_PR: process(LCLK, nUSB_TXE, nUSB_RXF)
247 begin
248
249 if(LCLK'event and LCLK = '1') then
250  nUSB_TXE_LCH <= nUSB_TXE;
251 end if;
252
253 if(LCLK'event and LCLK = '1') then
254  nUSB_RXF_LCH <= nUSB_RXF;
255 end if;
```

```
256
257 end process USB_FLAGS_PR;
258
259 --*************************************************************************************************************
260 -- read data from the USB chip
261 USB_BUS_PR: process (LCLK)
262 begin
263
264 if(LCLK'event and LCLK = '1') then
265  if(nUSB_RD = '0') then
266   USB_D_LCH0 <= USB_D;
267   USB_D_LCH1 <= USB_D_LCH0;
268   USB_D_LCH2 <= USB_D_LCH1;
269   USB_D_LCH3 <= USB_D_LCH2;
270  end if;
271 end if;
272
273 if(USB_WR = '1') then
274  USB_D <= LAD_IN;
275 else
276  USB_D <= (others => 'Z');
277 end if;
278
279 end process USB_BUS_PR;
280
281 --*************************************************************************************************************
282 -- local bus
283 LOCAL_BUS_LAD_PR: process (LCLK, nLRW, nLRDY)
284 begin
285
286 if(LCLK'event and LCLK = '1') then
287  if(STATE_MACHINE = local_lad) then
288   LAD_OUT <= USB_D_LCH1;   -- address
289  else
290   LAD_OUT <= USB_D_LCH0;   -- data
291  end if;
292 end if;
293
294 -- read data from internal registers
295 if(LCLK'event and LCLK = '1') then
296  if(STATE_MACHINE = local_data and nLRW = '0' and nLRDY = '0') then
297   LAD_IN <= LAD_INT;
298  end if;
299 end if;
300
301 end process LOCAL_BUS_LAD_PR;
302
303
304
305
306
```

```
--*****************************************************************************************************
-- local bus address strobe
LOCAL_BUS_ADS_PR: process (LCLK)
begin

if(LCLK'event and LCLK = '1') then
 if(STATE_MACHINE = local_lad) then
  nLADS <= '0';
 else
  nLADS <= '1';
 end if;
end if;

end process LOCAL_BUS_ADS_PR;

--*****************************************************************************************************
-- local bus chip-select
LOCAL_BUS_CS_PR: process (LCLK)
begin

-- chip-select for internal registers
if(LCLK'event and LCLK = '1') then
 if(STATE_MACHINE = local_data and USB_D_LCH2 = CS_INT_ADD) then
  nCSINT <= '0';
 else
  nCSINT <= '1';
 end if;
end if;

end process LOCAL_BUS_CS_PR;

--*****************************************************************************************************
-- AA - read command, 55 - write command
-- local bus nRW signal
LOCAL_BUS_RW_PR: process (LCLK)
begin

if(LCLK'event and LCLK = '1') then
 if(STATE_MACHINE = local_data and USB_D_LCH3 = READ_COMMAND) then
  nLRW <= '0';
 else
  nLRW <= '1';
 end if;
end if;

end process LOCAL_BUS_RW_PR;
```

```
--*****************************************************************************************************
-- internal registers access
--
INTERNAL_REG_PR: process (LCLK)
begin

-- local bus address latch
if(LCLK'event and LCLK = '1') then
 if(nLADS = '0') then
  PROC_ADD <= LAD_OUT;
 end if;
end if;

-- local bus wait states
if(LCLK'event and LCLK = '1') then
 if(nCSINT = '1') then
  WAIT_COUNTER <= "000";
 else
  WAIT_COUNTER <= WAIT_COUNTER + '1';
 end if;
end if;

-- local bus ready
if(LCLK'event and LCLK = '1') then
 if(nCSINT = '0' and PROC_ADD(7 downto 0) >= "00000000" and
    PROC_ADD <= "11111111" and WAIT_COUNTER = "100"
   ) then
   nLRDY <= '0';
 else
   nLRDY <= '1';
 end if;
end if;

-- local registers: write
if(LCLK'event and LCLK = '1') then
 if(nRESET = '0') then
  LED_REG <= (others => '0');
 elsif(nLRW = '1' and nCSINT = '0' and WAIT_COUNTER = "011") then
  if(PROC_ADD = LED_REG_ADD) then
   LED_REG <= LAD_OUT;                                          -- LED register (R/W)
  end if;
 end if;
end if;
```

```
409 --local registers: read
410 if(PROC_ADD = FPGA_REV_ADD) then
411  LAD_INT <= FPGA_REVISION;                    -- FPGA revision number register (R/O)
412 elsif(PROC_ADD = LED_REG_ADD) then
413  LAD_INT <= LED_REG;                          -- LED register (R/W)
414 else
415  LAD_INT <= (others => '0');
416 end if;
417
418 LED <= LED_REG;
419
420 end process INTERNAL_REG_PR;
421
422 end sample2_arch;
423
424
```