```
1   ----------------------------------------------------------------------------------
2   -- Company: Western Michigan University
3   -- Engineer: Erivelton Gualter dos Santos
4   --
5   -- Create Date:    21:50:25 06/03/2014
6   -- Design Name:
7   -- Module Name:    stopwatch - Behavioral
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 using
26  -- arithmetic functions with Signed or Unsigned values
27  --use IEEE.NUMERIC_STD.ALL;
28
29  -- Uncomment the following library declaration if instantiating
30  -- any Xilinx primitives in this code.
31  --library UNISIM;
32  --use UNISIM.VComponents.all;
33
34  entity stopwatch is
35     port (   CLK : in std_logic;
36              SW1 : in std_logic;
37              PB0 : in std_logic;
38              DP  : out std_logic;
39              AN  : out std_logic_vector (3 downto 0);
40              SEG : out std_logic_vector (6 downto 0));
41  end stopwatch;
42
43  architecture Behavioral of stopwatch is
44
45     signal CCLK : std_logic;
46     signal MCLK : std_logic;
47
48     signal SHIFTR1,SHIFTR2 : std_logic_vector (3 downto 0) := "0000";
49     signal DPB0 : std_logic := '0';
50     signal DSW1 : std_logic := '0';
51
52     type state is (IDLE, COUNT, STOP, FREEZE);
53     signal PRESENT_STATE : state:=IDLE;
54
55     signal DIG1: std_logic_vector (3 downto 0) := "0000";
56     signal DIG2: std_logic_vector (3 downto 0) := "0000";
57     signal DIG3: std_logic_vector (3 downto 0) := "0000";
```

```
58      signal DIG4: std_logic_vector (3 downto 0) := "0000";
59
60      signal BCD: std_logic_vector (3 downto 0);
61
62      signal COUNTING   : boolean := false;
63      signal RESET_COUNT: boolean := false;
64      signal DISPLAY_FREEZE: std_logic := '0';
65
66    begin
67
68    ------------------------------------------------
69    --------         Count Clock       ---------------
70    ------------------------------------------------
71    clk_main: process(CLK, CCLK)
72       variable DIVIDE : integer := 1000000;
73       variable COUNTER_DIV : integer :=0;
74    begin
75       if (rising_edge(CLK)) then
76          if(COUNTER_DIV < DIVIDE/2-1) then
77             COUNTER_DIV := COUNTER_DIV + 1;
78             CCLK <= '0';
79          elsif (COUNTER_DIV < DIVIDE-1) then
80             COUNTER_DIV := COUNTER_DIV + 1;
81             CCLK <= '1';
82          else
83             CCLK <= '0';
84             COUNTER_DIV := 0;
85          end if;
86       end if;
87    end process clk_main;
88
89    ------------------------------------------------
90    --------         Main Clock        ---------------
91    ------------------------------------------------
92    clk_segment: process(CLK, MCLK)
93       variable DIVIDE : integer := 1600;
94       variable COUNTER_DIV : integer :=0;
95    begin
96       if (rising_edge(CLK)) then
97          if(COUNTER_DIV < DIVIDE/2-1) then
98             COUNTER_DIV := COUNTER_DIV + 1;
99             MCLK <= '0';
100         elsif (COUNTER_DIV < DIVIDE-1) then
101            COUNTER_DIV := COUNTER_DIV + 1;
102            MCLK <= '1';
103         else
104            MCLK <= '0';
105            COUNTER_DIV := 0;
106         end if;
107      end if;
108   end process clk_segment;
109
110   ------------------------------------------------
111   --------      Debouncing SW1           -----------
112   ------------------------------------------------
113
114   process(MCLK,SW1)
```

```
115   begin
116      if (MCLK='1' and MCLK'event) then
117         SHIFTR1(2 downto 0) <= SHIFTR1(3 downto 1);
118         SHIFTR1(3) <= SW1;
119         if (SHIFTR1 = "0000") then
120            DSW1 <= '0';
121         else
122            DSW1 <= '1';
123         end if;
124      end if;
125      -- simulaco
126      -- DSW1 <= SW1;
127   end process;
128
129   ------------------------------------------------
130   --------      Debouncing PB0           -----------
131   ------------------------------------------------
132
133   process(MCLK,PB0)
134   begin
135      if (MCLK='1' and MCLK'event) then
136         SHIFTR2(2 downto 0) <= SHIFTR2(3 downto 1);
137         SHIFTR2(3) <= PB0;
138         if (SHIFTR2 = "0000") then
139            DPB0 <= '0';
140         else
141            DPB0 <= '1';
142         end if;
143      end if;
144      -- simulaco
145      -- DPB0 <= PB0;
146   end process;
147
148   ------------------------------------------------
149   --------      Display Controller       -----------
150   ------------------------------------------------
151
152   process(MCLK, DISPLAY_FREEZE, DIG1, DIG2, DIG3, DIG4)
153      variable SEL: STD_LOGIC_vector(1 downto 0) := "00";
154      variable FDIG1: std_logic_vector (3 downto 0) ;
155      variable FDIG2: std_logic_vector (3 downto 0) ;
156      variable FDIG3: std_logic_vector (3 downto 0) ;
157      variable FDIG4: std_logic_vector (3 downto 0) ;
158
159   begin
160      if (MCLK='1' and MCLK'event ) then
161         SEL:=SEL+1;
162      end if;
163      if (DISPLAY_FREEZE = '1' and DISPLAY_FREEZE'event) then
164         FDIG1 := DIG1;
165         FDIG2 := DIG2;
166         FDIG3 := DIG3;
167         FDIG4 := DIG4;
168      end if;
169
170      case SEL is
171         when "00" =>
```

```
172            if (DISPLAY_FREEZE = '1') then
173               BCD <= FDIG1;
174            else
175               BCD <= DIG1;
176            end if;
177            AN  <= "1110";
178            DP <= '1';
179         when "01"=>
180            if (DISPLAY_FREEZE = '1') then
181               BCD <= FDIG2;
182            else
183               BCD <= DIG2;
184            end if;
185            AN  <= "1101";
186            DP <= '1';
187         when "10"=>
188            if (DISPLAY_FREEZE = '1') then
189               BCD <= FDIG3;
190            else
191               BCD <= DIG3;
192            end if;
193            AN  <= "1011";
194            DP <= '0';
195         when others=>
196            if (DISPLAY_FREEZE = '1') then
197               BCD <= FDIG4;
198            else
199               BCD <= DIG4;
200            end if;
201            AN  <= "0111";
202            DP <= '1';
203      end case;
204
205   end process;
206
207   ----------------------------------------------------
208   --------     7 segments Decoder        -----------
209   ----------------------------------------------------
210   segment_process: process(BCD)
211   begin
212      case  BCD is
213      when "0000"=> SEG <="0000001";  -- '0'
214      when "0001"=> SEG <="1001111";  -- '1'
215      when "0010"=> SEG <="0010010";  -- '2'
216      when "0011"=> SEG <="0000110";  -- '3'
217      when "0100"=> SEG <="1001100";  -- '4'
218      when "0101"=> SEG <="0100100";  -- '5'
219      when "0110"=> SEG <="0100000";  -- '6'
220      when "0111"=> SEG <="0001111";  -- '7'
221      when "1000"=> SEG <="0000000";  -- '8'
222      when "1001"=> SEG <="0000100";  -- '9'
223      when others=> SEG <="1111111";
224      end case;
225   end process segment_process;
226
227   ----------------------------------------------------
228   --------          Counter          ---------------
```

```
229   ------------------------------------------------
230   counter: process(CCLK, DSW1, RESET_COUNT)
231
232   begin
233      if (CCLK'event and CCLK = '1') then
234         if (COUNTING = true) then
235            if (DIG4 = "0101" and DIG3 = "1001" and DIG2 = "1001" and DIG1 = "1001") then
       -- 59.99
236               DIG4 <= "0000";
237               DIG3 <= "0000";
238               DIG2 <= "0000";
239               DIG1 <= "0000";
240            elsif (DIG3 = "1001" and DIG2 = "1001" and DIG1 = "1001") then -- X9.99
241               DIG4 <= DIG4 + 1;
242               DIG3 <= "0000";
243               DIG2 <= "0000";
244               DIG1 <= "0000";
245            elsif (DIG2 = "1001" and DIG1 = "1001") then -- XX.99
246               DIG3 <= DIG3 + 1;
247               DIG2 <= "0000";
248               DIG1 <= "0000";
249            elsif (DIG1 = "1001") then -- XX.X9
250               DIG2 <= DIG2 + 1;
251               DIG1 <= "0000";
252            else -- XX.XX
253               DIG1 <= DIG1 + 1;
254            end if;
255         elsif (RESET_COUNT = true) then
256            DIG4 <= "0000";
257            DIG3 <= "0000";
258            DIG2 <= "0000";
259            DIG1 <= "0000";
260         end if;
261      end if;
262
263   end process counter;
264
265   ------------------------------------------------
266   --------         STATE MACHINE         -----------
267   ------------------------------------------------
268   machine: process(MCLK, DSW1, CCLK)
269      --variable reset: integer :=0;
270      --variable regreset: boolean := false;
271   begin
272      if (MCLK'event and MCLK = '1') then
273         case PRESENT_STATE is
274         when IDLE =>
275            DISPLAY_FREEZE <= '0';
276            COUNTING       <= false;
277
278            if (DPB0 = '1') then
279               PRESENT_STATE <= FREEZE;
280            elsif (DSW1 = '1') then
281               PRESENT_STATE <= COUNT;
282            else
283               PRESENT_STATE <= IDLE;
284            end if;
```

```
285
286         when COUNT =>
287            DISPLAY_FREEZE <= '0';
288            COUNTING       <= true;
289            RESET_COUNT    <= false;
290
291            if(DSW1 = '1' and DPB0 = '1') then
292               PRESENT_STATE <= FREEZE;
293            elsif (DSW1 = '1') then
294               PRESENT_STATE <= COUNT;
295            elsif (DSW1 = '0') then
296               PRESENT_STATE <= STOP;
297            end if;
298
299         when STOP =>
300            COUNTING       <= false;
301            RESET_COUNT    <= true;
302            DISPLAY_FREEZE <= '1';
303
304            if (DSW1 = '1') then
305               PRESENT_STATE <= COUNT;
306            else
307               PRESENT_STATE <= STOP;
308            end if;
309
310         when FREEZE =>
311            DISPLAY_FREEZE <= '1';
312            COUNTING       <= true;
313
314            if (DPB0 = '1') then
315               PRESENT_STATE <= FREEZE;
316            else
317               PRESENT_STATE <= COUNT;
318            end if;
319
320         end case;
321      end if;
322   end process machine;
323
324   end Behavioral;
325
326
```