```
1   ----------------------------------------------------------------------------------
2   -- Company: HSRW
3   -- Engineer: Andy Stamm
4   --
5   -- Create Date:    08:03:49 01/14/2016
6   -- Design Name:
7   -- Module Name:      Stopwatch
8   -- Project Name:     Lab 4.1
9   -- Target Devices: Xilinx Spartan 3
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  use IEEE.std_logic_signed.all;
25  use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if using
28  -- arithmetic functions with Signed or Unsigned values
29  --use IEEE.NUMERIC_STD.ALL;
30
31  -- Uncomment the following library declaration if instantiating
32  -- any Xilinx primitives in this code.
33  --library UNISIM;
34  --use UNISIM.VComponents.all;
35
36  entity Lab_4 is
37     port( clk: in std_logic;
38           BTN0, BTN3: in std_logic;
39           LED0, LED1: out std_logic:='0';
40           COM0,SEGA0,SEGB0,SEGC0,SEGD0,SEGE0,SEGF0,SEGG0,COM1,SEGA1,SEGB1,SEGC1,SEGD1,
    SEGE1,SEGF1,SEGG1: out STD_LOGIC:='0'
41           );
42
43  end Lab_4;
44
45  architecture Behavioral of Lab_4 is
46  signal clk_slow: std_logic;   -- slow clock signal
47  signal digit0, digit1, number: integer;            -- generate counting signal
48  signal d0, d1: std_logic_vector(6 downto 0);       -- vectors for the 7-segment display
49
50  begin
51     -- generate slow clock using the internal clock clk
52     -- slow_clk provides one tick every 50MHz
53     process is
54        variable count: integer range 0 to 50000001;
55        begin
56           if rising_edge(clk) then
```

```
57              clk_slow <= '0';        -- reset clk_slow to zero
58              count := count +1;      -- increment clock counter up to 50,000,000
59              if (count >= 5000000) then
60                 clk_slow <= '1';     -- set clk_slow to one for one cycle of clk
61                 count := 0;          -- reset clock counter to zero
62              end if;
63           end if;
64     end process;
65
66     -- generate integer counter
67     -- enable represents watch is on/off
68     -- startstop represents on/off mechanism of one button
69     process is
70     variable enable: std_logic:='0';
71     variable startstop: std_logic:='0';
72     begin
73
74        --Check for start/stop button and start/stop condition
75        if BTN0='1' then
76           startstop := '1';          -- set startstop to one if start button is pushed
77        elsif startstop='1' and rising_edge(clk_slow) then
78           startstop := '0';          -- reset startstop if startstop is enabled and
    clk_slow is one
79           if enable='0' then
80              enable:=  '1';          -- set enable to one startstop nd clk_slow is on
81           else
82              enable:='0';            -- reset enable in all other cases
83           end if;
84        end if;
85
86        -- Count if enabled and not Stop button pressed OR if start button pressed
87        -- if number > 99 then count for blinking output
88        if ( enable='1' and startstop='0') or (BTN0='1' and enable='0') or (number > 99)
     then
89           if rising_edge(clk_slow) then
90              if number < 1000 then --overflow protection, Reset to 100 for endless
    blinking
91                 number <= number + 1;
92              else
93                 number <= 100;
94              end if;
95           end if;
96        end if;
97
98        -- Check for reset button
99        if (BTN3='1') then
100             number <= 0;      -- reset counter
101             enable:='0';      -- reset enable
102       end if;
103
104       --debug LEDs
105       LED0<=enable;
106       LED1<=clk_slow;
107
108
109    end process;
110
```

```
111      -- mapping the integer to the two bit vectors
112      process is
113      begin
114
115      -- splitting the integer into two integer numbers
116      if number < 10 then
117      digit0 <= number;
118      digit1 <= 0;
119      elsif number < 20 then
120      digit0 <= number - 10;
121      digit1 <= 1;
122      elsif number < 30 then
123      digit0 <= number - 20;
124      digit1 <= 2;
125      elsif number < 40 then
126      digit0 <= number - 30;
127      digit1 <= 3;
128      elsif number < 50 then
129      digit0 <= number - 40;
130      digit1 <= 4;
131      elsif number < 60 then
132      digit0 <= number - 50;
133      digit1 <= 5;
134      elsif number < 70 then
135      digit0 <= number - 60;
136      digit1 <= 6;
137      elsif number < 80 then
138      digit0 <= number - 70;
139      digit1 <= 7;
140      elsif number < 90 then
141      digit0 <= number - 80;
142      digit1 <= 8;
143      elsif number < 100 then
144      digit0 <= number - 90;
145      digit1 <= 9;
146      elsif (number MOD 2) = 0 then
147      digit0 <= 10;
148      digit1 <= 10;
149      else
150      digit0 <= 9;
151      digit1 <= 9;
152      end if;
153
154      -- decoding digit0 to bit vector
155      if digit0 = 0 then
156      d0 <= "0111111";
157      elsif digit0 = 1 then
158      d0 <= "0000110";
159      elsif digit0 = 2 then
160      d0 <= "1011011";
161      elsif digit0 = 3 then
162      d0 <= "1001111";
163      elsif digit0 = 4 then
164      d0 <= "1100110";
165      elsif digit0 = 5 then
166      d0 <= "1101101";
167      elsif digit0 = 6 then
```

```
168       d0 <= "1111101";
169       elsif digit0 = 7 then
170       d0 <= "0000111";
171       elsif digit0 = 8 then
172       d0 <= "1111111";
173       elsif digit0 = 9 then
174       d0 <= "1101111";
175       elsif digit0 = 10 then
176       d0 <= "0000000";
177       end if;
178
179       -- decoding digit1 to bit vector
180       if digit1 = 0 then
181       d1 <= "0111111";
182       elsif digit1 = 1 then
183       d1 <= "0000110";
184       elsif digit1 = 2 then
185       d1 <= "1011011";
186       elsif digit1 = 3 then
187       d1 <= "1001111";
188       elsif digit1 = 4 then
189       d1 <= "1100110";
190       elsif digit1 = 5 then
191       d1 <= "1101101";
192       elsif digit1 = 6 then
193       d1 <= "1111101";
194       elsif digit1 = 7 then
195       d1 <= "0000111";
196       elsif digit1 = 8 then
197       d1 <= "1111111";
198       elsif digit1 = 9 then
199       d1 <= "1101111";
200       elsif digit1 = 10 then
201       d1 <= "0000000";
202       end if;
203       end process;
204
205       -- mapping the 7-segment displays
206       process is
207       begin
208       COM0 <= '0';
209       SEGA0<=d0(0);
210       SEGB0<=d0(1);
211       SEGC0<=d0(2);
212       SEGD0<=d0(3);
213       SEGE0<=d0(4);
214       SEGF0<=d0(5);
215       SEGG0<=d0(6);
216
217       COM1 <= '0';
218       SEGA1<=d1(0);
219       SEGB1<=d1(1);
220       SEGC1<=d1(2);
221       SEGD1<=d1(3);
222       SEGE1<=d1(4);
223       SEGF1<=d1(5);
224       SEGG1<=d1(6);
```

```
225       end process;
226
227    end Behavioral;
```