```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date: 11/18/2017 01:22:39 PM
6   -- Design Name:
7   -- Module Name: top - 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
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.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  --use IEEE.std_logic_arith.all;
30  --use IEEE.std_logic_unsigned.all;
31
32  -- Uncomment the following library declaration if instantiating
33  -- any Xilinx leaf cells in this code.
34  --library UNISIM;
35  --use UNISIM.VComponents.all;
36
37  entity top is
38    Port (
39      sw   : in std_logic_vector(7 downto 0);
40      btnR : in std_logic;
41      btnL : in std_logic;
42      clk  : in std_logic;
43
44      seg  : out std_logic_vector(6 downto 0);
45      dp   : out std_logic;
46      an   : out std_logic_vector(3 downto 0);
47      led  : out std_logic_vector(7 downto 0)
48    );
49  end top;
50
51  architecture Behavioral of top is
52
53      component encoder is
54          Port (
55                hex_in : in STD_LOGIC_VECTOR(3 DOWNTO 0);
56                a : out STD_LOGIC;
57                b : out STD_LOGIC;
58                c : out STD_LOGIC;
59                d : out STD_LOGIC;
60                e : out STD_LOGIC;
61                f : out STD_LOGIC;
62                g : out STD_LOGIC
63          );
64      end component encoder;
65
66
```

```
67      component ssd_muxer is
68          Port (
69              a_in        : in  std_logic_vector(3 downto 0);
70              b_in        : in  std_logic_vector(3 downto 0);
71              c_in        : in  std_logic_vector(3 downto 0);
72              d_in        : in  std_logic_vector(3 downto 0);
73              e_in        : in  std_logic_vector(3 downto 0);
74              f_in        : in  std_logic_vector(3 downto 0);
75              g_in        : in  std_logic_vector(3 downto 0);
76              decp0_in    : in  std_logic;
77              decp1_in    : in  std_logic;
78              decp2_in    : in  std_logic;
79              decp3_in    : in  std_logic;
80              seg_out     : out std_logic_vector(6 downto 0);
81              dp_out      : out std_logic;
82              an_out      : out std_logic_vector(3 downto 0);
83              clk         : in  STD_LOGIC
84          );
85      end component ssd_muxer;
86      signal a_in, b_in, c_in, d_in, e_in, f_in, g_in : std_logic_vector(3 downto 0) :=
        "0000";
87
88      component debounce is
89        Port (
90             CLK_100M : in std_logic;
91             SW       : in std_logic;
92             sglPulse : out std_logic;
93             Sig      : out std_logic);
94      end component debounce;
95      signal sglPulse : std_logic_vector(2 downto 0);
96      signal Sig : std_logic_vector(2 downto 0);
97
98      signal accum : std_logic_vector(7 downto 0);
99
100     --signal hundreds, tens, ones : unsigned(7 downto 0);
101
102     --signal bcd : std_logic_vector(11 downto 0) := "00";
103
104 begin
105
106     led <= sw;
107
108     DEBR: debounce port map (
109         CLK_100M => clk,
110         SW => btnR,
111         sglPulse => sglPulse(0),
112         Sig => Sig(0)
113     );
114
115     DEBL: debounce port map (
116         CLK_100M => clk,
117         SW => btnL,
118         sglPulse => sglPulse(1),
119         Sig => Sig(1)
120     );
121
122     -- set up all the connectors
123     -- A true master starts at 0 (who am I kidding?)
124     ENC0: encoder port map (
125         hex_in => accum(3 downto 0),
126         a => a_in(0),
127         b => b_in(0),
128         c => c_in(0),
129         d => d_in(0),
130         e => e_in(0),
131         f => f_in(0),
```

```
132             g => g_in(0)
133         );
134 
135         ENC1: encoder port map (
136             hex_in => accum(7 downto 4),
137             a => a_in(1),
138             b => b_in(1),
139             c => c_in(1),
140             d => d_in(1),
141             e => e_in(1),
142             f => f_in(1),
143             g => g_in(1)
144         );
145 
146     --    ENC2: encoder port map (
147     --        hex_in => bcd(3 downto 0),
148     --        a => a_in(0),
149     --        b => b_in(0),
150     --        c => c_in(0),
151     --        d => d_in(0),
152     --        e => e_in(0),
153     --        f => f_in(0),
154     --        g => g_in(0)
155     --    );
156 
157         a_in(2) <= '0';
158         b_in(2) <= '0';
159         c_in(2) <= '0';
160         d_in(2) <= '0';
161         e_in(2) <= '0';
162         f_in(2) <= '0';
163         --g_in(2) <= '0';
164         a_in(3) <= '0';
165         b_in(3) <= '0';
166         c_in(3) <= '0';
167         d_in(3) <= '0';
168         e_in(3) <= '0';
169         f_in(3) <= '0';
170         g_in(3) <= '0';
171 
172 
173         MUX: ssd_muxer port map (
174             a_in => a_in,
175             b_in => b_in,
176             c_in => c_in,
177             d_in => d_in,
178             e_in => e_in,
179             f_in => f_in,
180             g_in => g_in,
181 
182             decp0_in => '0',
183             decp1_in => '0',
184             decp2_in => '0',
185             decp3_in => '0',
186 
187             seg_out => seg,
188             dp_out => dp,
189             an_out => an,
190 
191             clk => clk
192         );
193 
194         --bcd <= conv_std_logic_vector(hundreds, 4) & conv_std_logic_vector(tens, 4) &
            conv_std_logic_vector(ones, 4);
195 
196         LOGIC: process (accum, sw, Sig)
```

```
197             variable counter : unsigned(3 downto 0) := "0000";
198             variable temp    : std_logic_vector(7 downto 0) := "00000000";
199         begin
200             if (Sig(1) = '1') then -- if we hit the accumulate button (button left)
201                 if (sw(7) = '1') then -- we got a negative value from out switches
202                     for counter in 0 to 7 loop
203                         if (sw(counter) = '0') then
204                             temp(counter) := '1';
205                         else
206                             temp(counter) := '0';
207                         end if;
208                     end loop;
209                     accum <= std_logic_vector(unsigned(temp) + 1); -- convert to insiged so
                        we can use + overload
210                     g_in(2) <= '1';
211                 else -- not negative
212                     accum <= sw;
213                     g_in(2) <= '0';
214                 end if;
215             else
216                 accum <= accum; -- store old value
217             end if;
218
219             if (Sig(0) = '1') then -- if Sig 2 is on then we want to reset the accum
220                 accum <= "00000000";
221                 g_in(2) <= '0';
222             end if;
223         end process LOGIC;
224
225
226     end Behavioral;
227
```