```vhdl
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 11/16/2017 02:45:02 PM
-- Design Name:
-- Module Name: top - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity top is
  Port (
         sw  : in  std_logic_vector(15 downto 0);
         clk : in  std_logic;
         seg : out std_logic_vector(6 downto 0);
         dp  : out std_logic;
         an  : out std_logic_vector(3 downto 0);
         led : out std_logic_vector(15 downto 0) );
end top;

architecture Behavioral of top is
    component ssd_muxer is
        Port (
            a_in       : in  std_logic_vector(3 downto 0);
            b_in       : in  std_logic_vector(3 downto 0);
            c_in       : in  std_logic_vector(3 downto 0);
            d_in       : in  std_logic_vector(3 downto 0);
            e_in       : in  std_logic_vector(3 downto 0);
            f_in       : in  std_logic_vector(3 downto 0);
            g_in       : in  std_logic_vector(3 downto 0);
            decp0_in   : in  std_logic;
            decp1_in   : in  std_logic;
            decp2_in   : in  std_logic;
            decp3_in   : in  std_logic;
            seg_out    : out std_logic_vector(6 downto 0);
            dp_out     : out std_logic;
            an_out     : out std_logic_vector(3 downto 0);
            clk        : in  STD_LOGIC
        );
    end component;
    signal a_in, b_in, c_in, d_in, e_in, f_in, g_in : std_logic_vector(3 downto 0);
    signal seg_out : std_logic_vector(6 downto 0);
    signal dp_out : std_logic;
    signal an_out : std_logic_vector(3 downto 0);

    component encoder
        Port (
              hex_in : in  STD_LOGIC_VECTOR(3 DOWNTO 0);
              a      : out STD_LOGIC;
              b      : out STD_LOGIC;
              c      : out STD_LOGIC;
              d      : out STD_LOGIC;
              e      : out STD_LOGIC;
              f      : out STD_LOGIC;
              g      : out STD_LOGIC
        );
    end component;

    -- signal st, ed : integer := 0;

begin
    seg <= seg_out;
    dp <= dp_out;
    an <= an_out;
    led <= sw;

    SSD_MUX: ssd_muxer port map(
        a_in => a_in,
        b_in => b_in,
        c_in => c_in,
        d_in => d_in,
        e_in => e_in,
        f_in => f_in,
        g_in => g_in,

        decp0_in => '0',
        decp1_in => '0',
        decp2_in => '0',
        decp3_in => '0',

        seg_out => seg_out,
        dp_out => dp_out,
        an_out => an_out,

        clk => clk
    );

--        -- https://www.ics.uci.edu/~jmoorkan/vhdlref/generate.html
--    REG_ENC: for I in 0 to 3 generate
--        --
https://stackoverflow.com/questions/10375858/how-to-slice-an-std-logic-vector-in-vhdl
--        REG_PMP: encoder port map(
--            hex_in => sw((I * 4) downto (((I + 1)) * 4 - 1)),
--            a => a_in(I),
--            b => b_in(I),
--            c => c_in(I),
--            d => d_in(I),
--            e => e_in(I),
--            f => f_in(I),
--            g => g_in(I)
--        );
--    end generate;


    REGE1: encoder port map(
        hex_in => sw(15 downto 12),
        a => a_in(3),
        b => b_in(3),
        c => c_in(3),
        d => d_in(3),
        e => e_in(3),
        f => f_in(3),
        g => g_in(3)
    );

    REGE2: encoder port map(
        hex_in => sw(11 downto 8),
        a => a_in(2),
        b => b_in(2),
        c => c_in(2),
        d => d_in(2),
        e => e_in(2),
        f => f_in(2),
        g => g_in(2)
    );

    REGE3: encoder port map(
        hex_in => sw(7 downto 4),
        a => a_in(1),
        b => b_in(1),
        c => c_in(1),
        d => d_in(1),
        e => e_in(1),
        f => f_in(1),
        g => g_in(1)
    );

    REGE4: encoder port map(
        hex_in => sw(3 downto 0),
        a => a_in(0),
        b => b_in(0),
        c => c_in(0),
        d => d_in(0),
        e => e_in(0),
        f => f_in(0),
        g => g_in(0)
    );

end Behavioral;

```