```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity leading_ones_counter is
   generic (
      BITS_IN: positive := 8;
      BITS_OUT: positive := 4);
   port (
      inp_vector: in std_logic_vector(BITS_IN-1 downto 0);

      -- the seven segment display characters are encoded in 7 bit values
      ssd: out std_logic_vector(6 downto 0));
end entity;


architecture concurrent of leading_ones_counter is
   signal lead_ones_cnt: std_logic_vector(BITS_OUT-1 downto 0);
   signal non_lead_ones_removed:  std_logic_vector(BITS_IN-1 downto 0);

   type integer_array is array (0 to BITS_IN) of integer range 0 to BITS_IN;
   signal lead_ones_inc_cnt: integer_array;

begin

   -- first remove all the non leading 1's from the input vector
   non_lead_ones_removed(BITS_IN-1) <= inp_vector(BITS_IN-1);
   gen1: for i in BITS_IN-2 downto 0 generate
      non_lead_ones_removed(i) <= non_lead_ones_removed(i+1) and inp_vector(i);
   end generate;

   -- then count the total number of leading ones and store that number
   --    in the MSB of leading_ones_cnt
   lead_ones_inc_cnt(0) <= 0;
   gen2: for i in 1 to BITS_IN generate
      lead_ones_inc_cnt(i) <= lead_ones_inc_cnt(i-1) + 1 when non_lead_ones_removed(i-1)
else lead_ones_inc_cnt(i-1);
   end generate;

   -- finally convert the leading ones count, stored in the MSB of leading_ones_inc_cnt, to
   --    an unsigned integer and then to a standard logic vector
   -- and store it in lead_ones_cnt
   lead_ones_cnt <= std_logic_vector(to_unsigned(lead_ones_inc_cnt(BITS_IN), BITS_OUT));

   -- then select the ssd character
   with lead_ones_cnt select
      ssd <= "0000001" when "0000",  -- 0
             "1001111" when "0001",  -- 1
             "0010010" when "0010",  -- 2
             "0000110" when "0011",  -- 3
             "1001100" when "0100",  -- 4
             "0100100" when "0101",  -- 5
             "0100000" when "0110",  -- 6
             "0001111" when "0111",  -- 7
             "0000000" when "1000",  -- 8
             "1111110" when others;

end architecture;
```