1 **library** ieee**;**

2 **use** ieee**.**std\_logic\_1164**.all;**

3 **use** ieee**.**numeric\_std**.all;**

4 **use** ieee**.**math\_real**.all;**

5 --------------------------------------------------------------------------------

6 **entity** bit\_counter **is**

7 **generic** **(**

8 BITS**:** positive **:=** 8 **);**

9 **port** **(**

10 x**:** **in** std\_logic\_vector**(**BITS**-**1 **downto** 0**);**

11 y**:** **out** std\_logic\_vector**(**integer**(**ceil**(**log2**(**real**(**BITS**))))** **downto** 0**);**

12 ssd**:** **out** std\_logic\_vector**(**6 **downto** 0**)** **);**

13 **end** **entity;**

14 --------------------------------------------------------------------------------

15 **architecture** leading\_ones\_counter **of** bit\_counter **is**

16 **type** int\_array **is** **array** **(**BITS**-**1 **downto** 0**)** **of** integer**;**

17 **signal** num\_ones**:** integer**;**

18 **begin**

19 **process(**x**)**

20 **variable** ones\_count**:** integer **range** 0 **to** BITS**;**

21 **begin**

22 ones\_count **:=** 0**;**

23 **for** i **in** x'**range** **loop**

24 **case** x**(**i**)** **is**

25 **when** '1' **=>** ones\_count **:=** ones\_count **+** 1**;**

26 **when** **others** **=>** **exit;**

27 **end** **case;**

28 **end** **loop;**

29 num\_ones **<=** ones\_count**;**

30 **end** **process;**

31

32 -- Convert result to slv output

33 y **<=** std\_logic\_vector**(to\_unsigned(**num\_ones**,** y'**length));**

34

35 -- Convert to SSD output

36 **with** num\_ones **select**

37 ssd **<=** "0000001" **when** 0**,** --"0" on SSD

38 "1001111" **when** 1**,** --"1" on SSD

39 "0010010" **when** 2**,** --"2" on SSD

40 "0000110" **when** 3**,** --"3" on SSD

41 "1001100" **when** 4**,** --"4" on SSD

42 "0100100" **when** 5**,** --"5" on SSD

43 "0100000" **when** 6**,** --"6" on SSD

44 "0001111" **when** 7**,** --"7" on SSD

45 "0000000" **when** 8**,** --"8" on SSD

46 "0000100" **when** 9**,** --"9" on SSD

47 "1100010" **when** **others;** --"o" for overflow

48 **end** **architecture;**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

use ieee.math\_real.all;

--------------------------------------------------------------------------------

entity bit\_counter is

  generic (

       BITS: positive := 8 );

port (

       x: in std\_logic\_vector(BITS-1 downto 0);

       y: out std\_logic\_vector(integer(ceil(log2(real(BITS)))) downto 0);

       ssd: out std\_logic\_vector(6 downto 0) );

end entity;

--------------------------------------------------------------------------------

architecture leading\_ones\_counter of bit\_counter is

type int\_array is array (BITS-1 downto 0) of integer;

   signal num\_ones: integer;

begin

process(x)

variable ones\_count: integer range 0 to BITS;

begin

ones\_count := 0;

for i in x'range loop

case x(i)  is

when '1' => ones\_count := ones\_count + 1;

when others => exit;

end case;

end loop;

num\_ones <= ones\_count;

end process;

y <= std\_logic\_vector(to\_unsigned(num\_ones, y'length));

   -- Convert to SSD output

   with num\_ones select

       ssd <= "0000001" when 0,       --"0" on SSD

               "1001111" when 1,       --"1" on SSD

               "0010010" when 2,       --"2" on SSD

               "0000110" when 3,       --"3" on SSD

               "1001100" when 4,       --"4" on SSD

               "0100100" when 5,       --"5" on SSD

               "0100000" when 6,       --"6" on SSD

               "0001111" when 7,       --"7" on SSD

               "0000000" when 8,       --"8" on SSD

               "0000100" when 9,       --"9" on SSD

               "1100010" when others;  --"o" for overflow

end architecture;