```
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 I_CELLS_NETS is
    port (
        elemento : in std_logic_vector(4 downto 0);
        red : out std_logic_vector(7 downto 0);
        BSC : out std_logic_vector(7 downto 0)
    );
end I_CELLS_NETS;

architecture Behavioral of I_CELLS_NETS is

    type mem is array ( 0 to 18) of std_logic_vector(7 downto
0);
     -- RED correspondiente a cada IN_CELL
     constant my_NET : mem := (
           0  => "00000001",
           1  => "00000010",
           2  => "00000011",
           3  => "00000100",
           4  => "00000101",
           5  => "00000110",
           6  => "00000111",
           7  => "00001000",
           8  => "00001001",
           9  => "00001010",
           10 => "00001011",
           11 => "00001011",
           12 => "00001100",
           13 => "00001100",
           14 => "00001101",
           15 => "00001101",
           16 => "00001110",
         17 => "00001110",
         18 => "00001110"
     );
     -- BSC sobre los 151 del total
     constant my_BSC : mem := (
            0  => "00000111",
            1  => "00000010",
            2  => "00000000",
            3  => "00110001",
            4  => "01101101",
            5  => "10010101",
            6  => "10001100",
            7  => "00011111",
```

```
        8  => "00011100",
        9  => "00011001",
        10 => "01001100",
        11 => "01111000",
        12 => "01000001",
        13 => "01110100",
        14 => "00111110",
        15 => "01110010",
        16 => "00101011",
        17 => "00101001",
        18 => "00100110"
    );
  begin

  process (elemento)
begin
    case elemento is
        when "00000" =>
            red <= my_NET(0);
            BSC <= my_BSC(0);
        when "00001" =>
            red <= my_NET(1);
            BSC <= my_BSC(1);
        when "00010" =>
            red <= my_NET(2);
            BSC <= my_BSC(2);
        when "00011" =>
            red <= my_NET(3);
            BSC <= my_BSC(3);
        when "00100" =>
            red <= my_NET(4);
            BSC <= my_BSC(4);
        when "00101" =>
            red <= my_NET(5);
            BSC <= my_BSC(5);
        when "00110" =>
            red <= my_NET(6);
            BSC <= my_BSC(6);
        when "00111" =>
            red <= my_NET(7);
            BSC <= my_BSC(7);
        when "01000" =>
            red <= my_NET(8);
            BSC <= my_BSC(8);
        when "01001" =>
            red <= my_NET(9);
            BSC <= my_BSC(9);
        when "01010" =>
            red <= my_NET(10);
            BSC <= my_BSC(10);
        when "01011" =>
            red <= my_NET(11);
            BSC <= my_BSC(11);
        when "01100" =>
            red <= my_NET(12);
            BSC <= my_BSC(12);
        when "01101" =>
```

```
                red <= my_NET(13);
                BSC <= my_BSC(13);
            when "01110" =>
                red <= my_NET(14);
                BSC <= my_BSC(14);
            when "01111" =>
                red <= my_NET(15);
                BSC <= my_BSC(15);
            when "10000" =>
                red <= my_NET(16);
                BSC <= my_BSC(16);
            when "10001" =>
                red <= my_NET(17);
                BSC <= my_BSC(17);
            when "10010" =>
                red <= my_NET(18);
                BSC <= my_BSC(18);
            when others =>
                red <= "00000000";
                BSC <= "00000000";
        end case;
    end process;

end Behavioral;
```