```
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 C_CELLS_NETS is
    port (
        elemento : in std_logic_vector(4 downto 0);
        BSC : out std_logic_vector(7 downto 0);
        CTRL : out std_logic_vector(7 downto 0);
        VALUE : out std_logic
    );
end C_CELLS_NETS;

architecture Behavioral of C_CELLS_NETS is

    type mem is array ( 0 to 18) of std_logic_vector(7 downto
0);

      -- 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"
    );
    -- CTRL cell correspondiente
    constant my_CTRL : mem := (
        0  => "00001000",
        1  => "00000011",
        2  => "00000001",
        3  => "00110010",
        4  => "01101110",
        5  => "10010110",
```

```
    6  => "10001101",
    7  => "00100000",
    8  => "00011101",
    9  => "00011010",
    10 => "01001101",
    11 => "01111001",
    12 => "01000010",
    13 => "01110101",
    14 => "00111111",
    15 => "01110011",
    16 => "00101100",
    17 => "00101010",
    18 => "00100111"
);

 begin

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

```
            when "01100" =>
                CTRL <= my_CTRL(12);
                BSC <= my_BSC(12);
            when "01101" =>
                CTRL <= my_CTRL(13);
                BSC <= my_BSC(13);
            when "01110" =>
                CTRL <= my_CTRL(14);
                BSC <= my_BSC(14);
            when "01111" =>
                CTRL <= my_CTRL(15);
                BSC <= my_BSC(15);
            when "10000" =>
                CTRL <= my_CTRL(16);
                BSC <= my_BSC(16);
            when "10001" =>
                CTRL <= my_CTRL(17);
                BSC <= my_BSC(17);
            when "10010" =>
                CTRL <= my_CTRL(18);
                BSC <= my_BSC(18);
            when others =>
                CTRL <= "00000000";
                BSC <= "00000000";
        end case;
    end process;

end Behavioral;
```