```
library ieee;
use ieee.std_logic_1164.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use ieee.numeric_std.all;
library lib_nanoproc;
use lib_nanoproc.all;
USE lib_nanoproc.nano_pkg.all;

-- Systeme nanoprocesseur
-- Ecole PHELMA - Grenoble INP
-- Auteur : Vincent FRISTOT
-- date : Mai 2008
--

entity SYSTEM_PROC is
PORT (  Resetn, Clock   : in std_logic;
    io_inp      : in std_logic_vector(31 downto 0);
    io_out      : out std_logic_vector(31 downto 0));
end SYSTEM_PROC;

-- systeme a nanoproc
-- 1 nanoproc
-- plan memoire pour bus d'adresses de 8 bits :
-- 0x00 - 0x7F : ROM
-- 0x80 - 0xBF : RAM, decodage sur A7A6
-- 0xC0      - 0xFF : decodage registres I/O, poids faible A0=0

architecture a of SYSTEM_PROC is

component PROC
port (  din          : in std_logic_vector(len_data_bus-1 downto 0);
    resetn, clock    : in std_logic;
    write        : out std_logic;
    data_bus     : out std_logic_vector(len_data_bus-1 downto 0);
    addr_bus     : out std_logic_vector(len_addr_bus-1 downto 0));
end component;

component RAM
generic (add_bits   : integer;
     data_bits  : integer);
port (
         add         : in std_logic_vector(add_bits-1 downto 0);
     data_in    : in std_logic_vector(data_bits-1 downto 0);
     data_out   : out std_logic_vector(data_bits-1 downto 0);
     clk,write  : in std_logic);
end component;

component ROM
port (  address      : in std_logic_vector(len_addr_bus-2 downto 0) ;
    output       : out std_logic_vector(15 downto 0) ) ;
end component;

component PORT_IO
port (  clk, reset_n, ecr : in std_logic;
    adress       : in std_logic;
    dat_in_io    : in std_logic_vector(len_data_bus-1 downto 0);
    in_io        : in std_logic_vector(31 downto 0);
    dat_out_io      : out std_logic_vector(len_data_bus-1 downto 0);
    out_io      : out std_logic_vector(31 downto 0)) ;
end component;

signal          w_ram,wr,ecr_port  :  std_logic;
signal          d_in,data_ram,data_rom,data_in,data_port  :  std_logic_vector(len_data_bus-1
downto 0);
signal          sys_add_bus     : std_logic_vector(len_addr_bus-1 downto 0);
signal      sys_add_ram : std_logic_vector(len_addr_bus-3 downto 0);
signal      sys_add_rom : std_logic_vector(len_addr_bus-2 downto 0);
signal      sys_add_port    : std_logic;

begin

U1:proc port map(
           din=>d_in,
           Resetn=>Resetn,
           Clock=>Clock,
           write=>wr,
           data_bus=>data_in,
           addr_bus =>sys_add_bus);

U2:ram  generic map (len_addr_bus-2,len_data_bus) -- Adresses sur 6 bits
    port map(
        add=>sys_add_ram,
        data_in=>data_in,
        data_out=>data_ram,
        clk=>Clock,
        write=>w_ram);

U3: rom port map(                -- adresses sur 7 bits
        address=> sys_add_rom,
        output=> data_rom);

U4 : port_io port map(
        clk => Clock,
        reset_n=>Resetn,
        ecr => ecr_port,
        adress => sys_add_port,
        dat_in_io => data_in,
        in_io => io_inp,
        dat_out_io => data_port,
        out_io => io_out);


process (sys_add_bus,wr,data_rom,data_ram,data_port)
begin
    ecr_port<='0';
    w_ram<='0';
    d_in<=(others=>'0');
    sys_add_ram <= sys_add_bus(len_addr_bus-3 downto 0);      -- bus adresses pour RAM
    sys_add_port<= sys_add_bus(0);
    sys_add_rom <= sys_add_bus(len_addr_bus-2 downto 0);

    case sys_add_bus(7 downto 6) is           -- decodage adresses sur A7A6
    when "10" =>                       -- decodage RAM
        if (wr='1' ) then w_ram<='1'; end if;     -- decodage sur adresse A7 pour SRAM
        d_in<=data_ram;
    when "11" =>                    -- decodage port I/O
        if (wr='1') then ecr_port<='1'; end if;
        d_in<=data_port;
    when others =>                  -- decodage ROM
        d_in<=data_rom;
    end case;
end process;

end a;
```