```
Nbin <= conv_std_logic_vector(N-1,address);
t9: contador generic map (address) port map (clock,rst5,ha6,addr_re);
t10: comparador generic map (address) port map (addr_re,Nbin,sfin);
t11: contador generic map (2) port map (clock,rst0,ha0,n11);
t12: comparador generic map (2) port map (n11,"01",fmux3);
t13: mux3 generic map (data) port map (x_mari,y_mari,z_mari,n11,out_m);
t14: ram generic map (address,data) port map (clock,out_m,addr_wr,addr_re,we2,fft_out);
t15: contador generic map (2) port map (clock,rst6,ha7,n15);
t16: comparador generic map (2) port map (n15,"11",fin3);
end estructura;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

package split_package is
constant N : integer := 32;
constant address : integer := 5; --logaritmo en base 2 de N
constant bits : integer := 16;
constant data : integer := 32;
type arreglo is array(0 to 3) of std_logic_vector(address-1 DOWNTO 0);
type arreglo1 is array(0 to 1) of std_logic_vector(data-1 DOWNTO 0);
type arreglo2 is array(0 to 3) of std_logic_vector(data-1 DOWNTO 0);
type arreglo3 is array(0 to 2) of std_logic_vector(data-1 DOWNTO 0);
component div2 is
   generic(bits : positive := 10); -- número de bits de los datos
   port (clk,sel,hab:in std_logic;
         y:in std_logic_vector(bits-1 downto 0);
         Q:out std_logic_vector(bits-1 downto 0));
end component;
component comparador is
  generic (bits : integer := 10);
  port (cuenta,tope: in std_logic_vector(bits-1 downto 0);
          fin: out std_logic);
end component;
component mux1 is
  generic(bits : positive := 10);
  port (a,b: in std_logic_vector (bits-1 downto 0);
          sel: in std_logic;
          y: out std_logic_vector (bits-1 downto 0));
end component;
component contador is
   generic(bits : positive := 10);
   port (clk,reset,hab:in std_logic;
           Q:out std_logic_vector(bits-1 downto 0));
end component;
component mux2 is
   generic(bits : positive := 10);
```

```
  port (a,b,c,d: in std_logic_vector (bits-1 downto 0);
        sel: in std_logic_vector (1 downto 0);
        y: out std_logic_vector (bits-1 downto 0));
end component;
component div4 is
   generic(bits : positive := 10); -- número de bits de los datos
   port (clk,sel,hab:in std_logic;
         y:in std_logic_vector(bits-1 downto 0);
         Q:out std_logic_vector(bits-1 downto 0));
end component;
component reg1 is
   port (clk,reset,hab,D:in std_logic;
         Q : out std_logic);
end component;
component mayor is
  generic(bits : integer := 10);
  port (clk: in std_logic;
         inicial: in std_logic_vector(bits-2 downto 0);
         data,ref: in std_logic_vector(bits-1 downto 0);
         reset: out std_logic;
         D: out std_logic_vector(bits-1 downto 0));
end component;
component dmux is
  generic (bits : integer := 10);
  port (y: in std_logic_vector (bits-1 downto 0);
        sel: in std_logic_vector (1 downto 0);
        z: out arreglo);
end component;
component reg is
   generic(bits : positive := 10);
   port (clk,reset,hab:in std_logic;
         D: in std_logic_vector (bits-1 downto 0);
         Q: out std_logic_vector (bits-1 downto 0));
end component;
component dmux_reg is
  generic (bits : integer := 10);
  port (clk,reset: in std_logic;
         y: in std_logic_vector (bits-1 downto 0);
         sel: in std_logic_vector (1 downto 0);
         Q0,Q1,Q2,Q3: out std_logic_vector (bits-1 downto 0));
end component;
component mux5 is
  generic(bits : positive := 10);
  port (a,b,c,d,e: in std_logic_vector (bits-1 downto 0);
        sel: in std_logic_vector (2 downto 0);
        y: out std_logic_vector (bits-1 downto 0));
end component;
component ram is
         generic(ADDRESS_WIDTH : integer := 10;
                 DATA_WIDTH : integer := 32);
         port (clock : IN  std_logic;
```

```
            data : IN  std_logic_vector(DATA_WIDTH - 1 DOWNTO 0);
            addr_wr,addr_re : IN  std_logic_vector(ADDRESS_WIDTH - 1 DOWNTO 0);
            we : IN  std_logic;
            q : OUT std_logic_vector(DATA_WIDTH - 1 DOWNTO 0));
end component;
component split_entrada is
generic(data_ram : positive := 32;
          bits : positive := 10;
          N : positive := 1024);
  port (clock : in std_logic;
entrada,a_mari,b_mari,c_mari,d_mari : in std_logic_vector(data_ram-1 downto 0);
reset0,hab0,sel1,sel2,hab1,stope,reset1,hab2,reset2,hab3,reset3,hab4,sel3,hab5 : in std_logic;
reset4,hab6,sel4,hab7,reset5,hab8,reset6,hab9,reset7,reset8,hab10,we1 : in std_logic;
sel5,reset9,hab11,reset10,reset11,reset12,hab12,reset13,hab13 : in std_logic;
sel6: in std_logic_vector(1 downto 0);
a_dato,b_dato,c_dato,d_dato,x_dato,y_dato,z_dato : out std_logic_vector(data_ram-1 downto 0);
pares,fcont0,fin,fcont2,fcont5,salto,mayor1,fcont13,fase,ini,fin_b : out std_logic);
end component;
component maq_entrada IS
   PORT (clock,reset_entrada,inicio1,par : IN STD_LOGIC;
pares,fcont0,fin,fcont2,fcont5,salto,mayor1,fcont13,fase,ini : in std_logic;
reset0,hab0,sel1,sel2,hab1,stope,reset1,hab2,reset2,hab3,reset3,hab4,sel3,hab5 : out std_logic;
reset4,hab6,sel4,hab7,reset5,hab8,reset6,hab9,reset7,reset8,hab10,we1 : out std_logic;
sel5,reset9,hab11,reset10,reset11,reset12,hab12,reset13,hab13 : out std_logic;
sel6: out std_logic_vector(1 downto 0));
END component;
component dmux1 is
  generic (bits : integer := 32);
  port (y: in std_logic_vector (bits-1 downto 0);
         sel: in std_logic;
         z: out arreglo1);
end component;
component dmux2 is
  generic (bits : integer := 32);
  port (y: in std_logic_vector (bits-1 downto 0);
         sel: in std_logic_vector (1 downto 0);
         z: out arreglo2);
end component;
component dmux_reg2 is
  generic (bits : integer := 32);
  port (clk,reset: in std_logic;
          y: in std_logic_vector (bits-1 downto 0);
          sel: in std_logic_vector (1 downto 0);
          Q0,Q1,Q2,Q3: out std_logic_vector (bits-1 downto 0));
end component;
component mul2 is
   generic(bits : positive := 10); -- número de bits de los datos
   port (clk,sel,hab:in std_logic;
         y:in std_logic_vector(bits-1 downto 0);
         Q:out std_logic_vector(bits-1 downto 0));
end component;
```

```
        f0,fmari,
        a_mari,b_mari,c_mari,d_mari,y_mari,z_mari);
t3: maq_mariposa port map
        (clock,reset_mariposa,inicio2,
        fcont0,f0,fmari,
        s0,h0,r1,h1,s1,h2,s2,h3,re0,en0,en1,en2,en3,re1,en4,re2,en5,en6);
t4: split_orden generic map (data,address,N) port map
        (clock,
        rst0,ha0,par1,rst1,ha1,rst2,ha2,rst3,ha3,arst4,ha4,ha5,rst5,ha6,we2,rst6,ha7,
        x_dato,y_mari,z_mari,
        fmux3,fmux4,fmux2,sfin,fin3,
        fft_out);
t5: maq_orden port map
        (clock,reset_orden,inicio3,par,fin,fmux3,fmux4,fmux2,sfin,
        rst0,ha0,par1,rst1,ha1,rst2,ha2,rst3,ha3,arst4,ha4,ha5,rst5,ha6,we2,rst6,ha7);
t6: maq_split port map
        (clock,reset,inicio,
        fcont2,fin_b,fmari,fcont0,fin3,fin,
        reset_entrada,inicio1,reset_mariposa,inicio2,reset_orden,inicio3);
end estructura;
```