Escola Politècnica Superior
d'Enginyeria de Manresa
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PRÀCTICA 3: DISSENY BANC REGISTRES I PC

Sergi Carol Bosch i Enric Lenard Uró

Grau en Enginyeria de Sistemés TIC

Arquitectura de Computadors

Curs 2013-14, Grup 10, G12 de pràctiques

Realització de la pràctica: 07/04/2014

Lliurament del treball: 28/04/2014

Realització pràctica

• L'objectiu d'aquesta practica és crear el banc de registres i el comptador de programa descrits a classe, per aquest motiu anirem realitzant un disseny incremental.
En primer lloc, començarem pel banc de registres, partint d'un senzill registre de 32 bits que anirem ampliant en la seva funcionalitat fins a la construcció del banc complet. Finalment, implementarem el comptador de programa i el mòduls necessaris per el correcte funcionament d'aquest.

Per tant, primer de tot dissenyem el modul Registre que tindrà les següents condicions:

- Un senyal d'habilitació o escriptura (enable) actiu a nivell alt.
- Un senyal de rellotge (clk) per determinar el moment de l'actualització que s'efectuar à amb el flanc de baixada del senyal de rellotge del processador
- Un senyal de reset asíncron i actiu a nivell alt.
- I finalment, les dades d'entrada (Din) i les de sortida (Dout).

REGISTRE

```
1   library ieee;                          --Cridem les llibreries necessaries.
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_unsigned.all;
4
5   entity Registre is
6      Port ( Reset : in STD_LOGIC;
7             clk : in STD_LOGIC;          --Senyal de rellotge.
8             enable : in STD_LOGIC;       --Senyal d'habilitacio.
9             Din : in STD_LOGIC_VECTOR(31 downto 0); --Dades d'entrada.
10            Dout : out STD_LOGIC_VECTOR(31 downto 0)); --Dades de sortida.
11  end Registre;
12
13  architecture Behavioral of Registre is
14  --Creem senyal per modificar valor del registre.
15  signal reg : std_logic_vector(31 downto 0):=x"00000000";
16  begin
17
18  process(clk,reset)
19  begin
20     if (reset = '1') then
21        reg <= x"00000000";  --Posem a zero el registre.
22     elsif (enable = '1') then
23        --Actuem en el flanc de baixada del senyal del rellotge.
24        if (falling_edge(clk)) then
25           reg <= Din;          --Posem la dada d'entrada al registre.
26        end if;
27     end if;
28  end process;
29  Dout <= reg;                  --Asociem el valor del registre a la sortida.
30
31  end Behavioral;
```

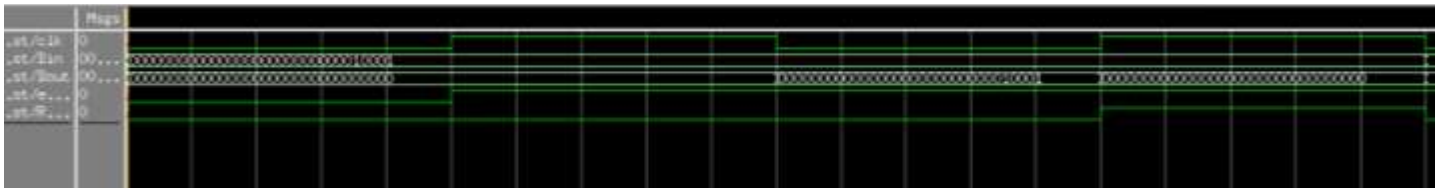
Un cop tenim el disseny del registre de 32 bits acabat realitzem una prova del circuit aplicant uns valors determinats per poder comprovar el seu correcte funcionament.

TEST BENCH REGISTRE

```
27  LIBRARY ieee;
28  USE ieee.std_logic_1164.all;
29
30  ENTITY Registre_vhd_tst IS
31  END Registre_vhd_tst;
32  ARCHITECTURE Registre_arch OF Registre_vhd_tst IS
33  -- constants
34  -- signals
35  SIGNAL clk : STD_LOGIC;
36  SIGNAL Din : STD_LOGIC_VECTOR(31 DOWNTO 0);
37  SIGNAL Dout : STD_LOGIC_VECTOR(31 DOWNTO 0);
38  SIGNAL enable : STD_LOGIC;
39  SIGNAL Reset : STD_LOGIC;
40  COMPONENT Registre
41      PORT (
42      clk : IN STD_LOGIC;
43      Din : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
44      Dout : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
45      enable : IN STD_LOGIC;
46      Reset : IN STD_LOGIC
47      );
48  END COMPONENT;
49  BEGIN
50      i1 : Registre
51      PORT MAP (
52  -- list connections between master ports and signals
53      clk => clk,
54      Din => Din,
55      Dout => Dout,
56      enable => enable,
57      Reset => Reset
58      );
59  init : PROCESS
60  -- variable declarations
61  BEGIN
62      clk<='0';
63      Reset<='0';
64      enable<='0';
65      Din<=x"00000011";
66      wait for 10 us;
67      clk<='1';
68      enable<='1';
69      wait for 10 us;
70      clk<='0';
71      wait for 10 us;
72      clk<='1';
73      Reset<='1';
74      wait for 10 us;
75      clk<='0';
76      Reset<='0';
77      Din<=x"11110000";
78      wait for 10 us;
79      clk<='1';
80      wait for 10 us;
81      clk<='0';
82      enable<='0';
83      Din<=x"00001111";
84      wait for 10 us;
85  WAIT;
86  END PROCESS init;
87  always : PROCESS
88  -- optional sensitivity list
89  -- (        )
90  -- variable declarations
91  BEGIN
92          -- code executes for every event on sensitivity list
93  WAIT;
94  END PROCESS always;
95  END Registre_arch;
```

Un cop tenim el programa de prova creem una gràfica que ens mostra el resultat visualment.

SIMULACIÓ GRÀFICA REGISTRE



A continuació, per connectar els registres de 32 bits amb els busos d'entrada i sortida creem el mòdul RegSortida3Estats.

Per realitzar el codi de la entitat RegSortida3Estats tenim dues opcions per triar:

1. Connectar totes les sortides dels registres a un gran multiplexor perquè deixi arribar als busos del processador (connectats a Sor1 i Sor2) el contingut dels registres seleccionats per les entrades rf1 i rf2. En aquest cas serien dos multiplexors que tindrien 32 entrades de 32 bits i una sortida de 32 bits,
   controlats per rf1 i rf2 cadascun.

2. Dotar als registres que s'acaben de definir de dues sortides de tres estats fent que en funció de si es vol llegir un registre determinat per algun dels dos busos s'activi la sortida corresponent. D'aquesta manera es pot connectar directament les sortides dels 32 registres als busos de sortida Sor1 i Sor2.

Nosaltres utilitzarem la segona opció , ja que es la que trobem més senzilla.
En aquet cas utilitzarem la entitat Registre explicada anteriorment per tal de controlar el clock i el reset. Així el RegSortida3Estats l'únic que farà serà crear un component de la entitat Registre ,si el senyals de habilitació de escriptura està WE està activat es realitzarà una operació de escriptura al registre.

A continuació es comprovarà si els valors de lectura del registre E1 i E2 estan activats. Si un o els dos valors està activat el valor de Din passarà a ser el de Dout, en cas contrari les sortides es trobaran en alta impedància (Z) .

RegSortida3Estats

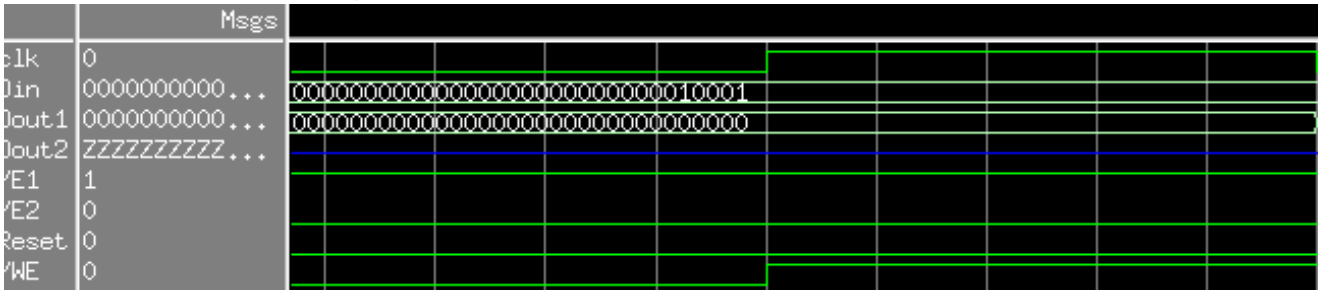
```
1  library ieee;                          --Cridem les llibreries necessaries.
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity RegSortida3Estats is
6     Port (
7        Reset : in STD_LOGIC;
8        clk : in STD_LOGIC;
9        WE : in STD_LOGIC;          --Senyal d'habilitacio d'escriptura.
10       Din : in STD_LOGIC_VECTOR (31 downto 0);
11       E1 : in STD_LOGIC;
12       E2 : in STD_LOGIC;
13       Dout1 : out STD_LOGIC_VECTOR (31 downto 0);  --Bus del cami de dades anomenat sor1.
14       Dout2 : out STD_LOGIC_VECTOR (31 downto 0)); --Bus del cami de dades anomenat sor2.
15 end RegSortida3Estats;
16
17 architecture Behavioral of RegSortida3Estats is
18    --Cridem els components dels altres arxius.
19    component Registre is
20       Port (
21          Reset : in STD_LOGIC;
22          clk : in STD_LOGIC;
23          Enable : in STD_LOGIC;
24          Din : in STD_LOGIC_VECTOR (31 downto 0);
25          Dout : out STD_LOGIC_VECTOR (31 downto 0));
26    end component;
27    --Creem una senyal de sortida per modificar els valors.
28    signal SortidReg : STD_LOGIC_VECTOR (31 downto 0);
29    begin
30    --Associem els diferents senyals de l'arxiu Registre amb aquets.
31    reg : Registre port map(
32       Reset => Reset,
33       clk => clk,
34       enable =>WE,
35       Din => Din,
36       Dout => SortidReg);
37    --Indiquem valor sortida segons E1 i E2.
38    Dout1 <= SortidReg when E1='1' else "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ";
39    Dout2 <= SortidReg when E2='1' else "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ";
40 end Behavioral;
```

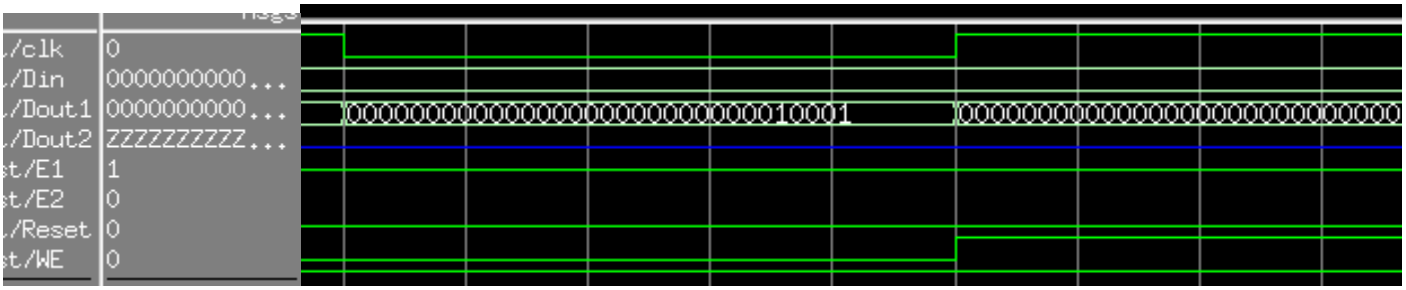
A continuació creem un test en el que anirem fent toggle al enable per veure com el valor canvia.

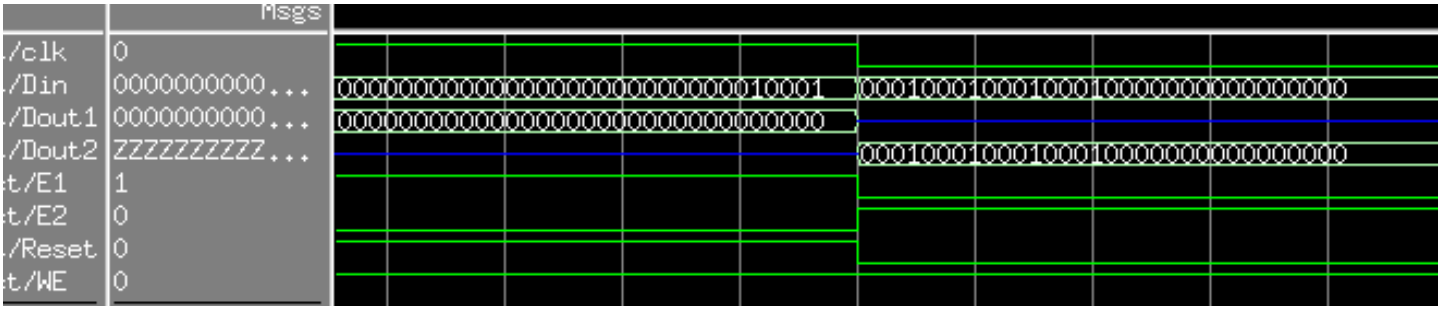
TEST BENCH RegSortida3Estats

```
70  BEGIN
71     clk<='0';
72     Reset<='0';
73     WE<='0';
74     Din<=x"00000011";
75     E1<='1';
76     E2<='0';
77     wait for 10 us;
78     clk<='1';
79     WE<='1';
80     wait for 10 us;
81     clk<='0';
82     wait for 10 us;
83     clk<='1';
84     Reset<='1';
85     wait for 10 us;
86     clk<='0';
87     Reset<='0';
88     Din<=x"11110000";
89     E1<='0';
90     E2<='1';
91     wait for 10 us;
92     clk<='1';
93     wait for 10 us;
94     clk<='0';
95     WE<='0';
96     Din<=x"00001111";
97     wait for 10 us;
98  WAIT;
```

SIMULACIÓ GRÀFICA RegSortida3Estats







A continuació vam realitzar la entitat RegZero.

El registre 0 és un registre especial en el que el seu valor no és pot canviar I sempre val 0. Aquet valor és independent del valor del clock, encara que estigui declarat, ja que no és pot sobreescriure. Com podem veure sempre té el valor de 0 i en el cas de que no es seleccioni la activació el valor serà de alta impedancia (Z).

REGISTRE ZERO

```
1   library ieee;                          --Cridem les llibreries necessaries.
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_unsigned.all;
4
5   entity RegistreZero is
6      port ( clk : in STD_LOGIC;
7             reset : in  STD_LOGIC;
8             E1 : in STD_LOGIC; --Indica on s'ha de deixar l'informacio del regist
9             E2 : in STD_LOGIC;
10            Dout1 : out STD_LOGIC_VECTOR(31 downto 0);
11            Dout2 : out STD_LOGIC_VECTOR(31 downto 0));
12  end RegistreZero;
13
14  architecture Zero of RegistreZero is
15  begin
16     --Indiquem valor sortida (0 o Z) segons E1 i E2.
17     Dout1 <= x"00000000" when E1='1' else "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ";
18     Dout2 <= x"00000000" when E2='1' else "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ";
19  end Zero;
```

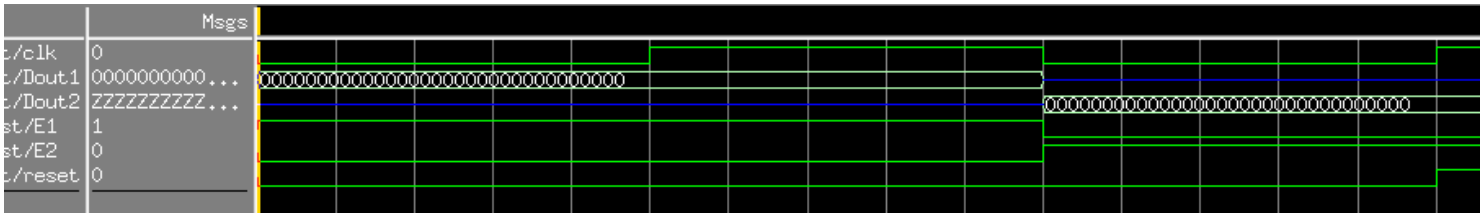
El registre 0 té un Test bench relativament senzill ja que no s'hi pot escriure i sempre té el mateix valor.

TEST BENCH REGISTRE ZERO

```
62  init : PROCESS
63  -- variable declarations
64  BEGIN
65     clk<='0';
66     E1<='1';
67     E2<='0';
68     reset<='0';
69     wait for 10 us;
70     clk<='1';
71     wait for 10 us;
72     clk<='0';
73     E1<='0';
74     E2<='1';
75     wait for 10 us;
76     clk<='1';
77     reset<='1';
78     wait for 10 us;
79  WAIT;
```

La gràfica resultant és la següent:

SIMULACIÓ GRÀFICA REGISTRE ZERO



Com podem veure el valor resultant sempre és 0.

Abans de parlar del Banc de Registres tenim que fer un descodificador.

El descodificador és l'encarregat de descodificar les entrades de 5 bits cap a sortides de 32 bits per tal de triar un registre i els enables de les sortides que es troben a la entitat RegSortida3Estats. Això es fa desplaçant un bit cap a l'esquerra cada cop que el numero de entrada s'incrementa en 1 fins que el bit mencionat arriba a la posició 32.

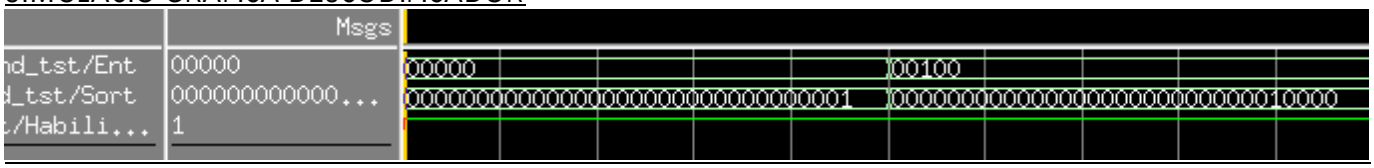
DESCODIFICADOR

```
library ieee;                              --Cridem les llibreries necessaries.
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity Descodificador5 is
   Port ( Habilitacio: in STD_LOGIC;
      Ent : in STD_LOGIC_VECTOR (4 downto 0);
      Sort : out STD_LOGIC_VECTOR (31 downto 0));
end Descodificador5;

architecture desc of Descodificador5 is
begin
   process (Habilitacio,Ent)
      begin
      if Habilitacio='1' then
      case Ent is
         when "00000" => Sort <= "00000000000000000000000000000001";
         when "00001" => Sort <= "00000000000000000000000000000010";
         when "00010" => Sort <= "00000000000000000000000000000100";
         when "00011" => Sort <= "00000000000000000000000000001000";
         when "00100" => Sort <= "00000000000000000000000000010000";
         when "00101" => Sort <= "00000000000000000000000000100000";
         when "00110" => Sort <= "00000000000000000000000001000000";
         when "00111" => Sort <= "00000000000000000000000010000000";
         when "01000" => Sort <= "00000000000000000000000100000000";
         when "01001" => Sort <= "00000000000000000000001000000000";
         when "01010" => Sort <= "00000000000000000000010000000000";
         when "01011" => Sort <= "00000000000000000000100000000000";
         when "01100" => Sort <= "00000000000000000001000000000000";
         when "01101" => Sort <= "00000000000000000010000000000000";
         when "01110" => Sort <= "00000000000000000100000000000000";
         when "01111" => Sort <= "00000000000000001000000000000000";
         when "10000" => Sort <= "00000000000000010000000000000000";
         when "10001" => Sort <= "00000000000000100000000000000000";
         when "10010" => Sort <= "00000000000001000000000000000000";
         when "10011" => Sort <= "00000000000010000000000000000000";
         when "10100" => Sort <= "00000000000100000000000000000000";
         when "10101" => Sort <= "00000000001000000000000000000000";
         when "10110" => Sort <= "00000000010000000000000000000000";
         when "10111" => Sort <= "00000000100000000000000000000000";
         when "11000" => Sort <= "00000001000000000000000000000000";
         when "11001" => Sort <= "00000010000000000000000000000000";
         when "11010" => Sort <= "00000100000000000000000000000000";
         when "11011" => Sort <= "00001000000000000000000000000000";
         when "11100" => Sort <= "00010000000000000000000000000000";
         when "11101" => Sort <= "00100000000000000000000000000000";
         when "11110" => Sort <= "01000000000000000000000000000000";
         when "11111" => Sort <= "10000000000000000000000000000000";
         when others => Sort <= "--------------------------------";
      end case;

   elsif (Habilitacio='0') then
      Sort <= x"00000000";
      end if;
   end process;
end desc;
```

TEST BENCH DESCODIFICADOR

```
53  init : PROCESS
54  -- variable declarations
55  BEGIN
56
57      Ent<="00000";
58      Habilitacio<='1';
59      wait for 10 us;
60      Ent<="00100";
61      wait for 10 us;
62      Ent<="01001";
63      Habilitacio<='0';
64      wait for 10 us;
65      Ent<="11111";
66      Habilitacio<='1';
67      wait for 10 us;
68
69  WAIT;
```

SIMULACIÓ GRÀFICA DESCODIFICADOR



Finalment creem la entitat Banc de Registres.

El banc de registres és el component que engloba totes les altres entitats. En si el banc de registre el que fa es descodificar les entrades rf1,rf2 i rdest, deixant les entrades lSor1 ,lSor2 i esc com a enables dels descodificadors anteriors respectivament.
Un cop descodificats els valors procedim a crear 32 registres utilitzant les entitats de RegistreZero i RegSortida3Estats i passant com a paràmetres els valors descodificats.

BANC DE REGISTRES

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

entity BancRegistres is
    Port ( Reset : in STD_LOGIC;
           clk : in STD_LOGIC;
           esc : in STD_LOGIC;
           rdest : in STD_LOGIC_VECTOR(4 downto 0);
           dades : in STD_LOGIC_VECTOR(31 downto 0);
           lSor1: in STD_LOGIC;
           rf1 : in STD_LOGIC_VECTOR(4 downto 0);
           lSor2: in STD_LOGIC;
           rf2 : in STD_LOGIC_VECTOR(4 downto 0);
           sor1 : out STD_LOGIC_VECTOR(31 downto 0);
           sor2 : out STD_LOGIC_VECTOR(31 downto 0));
end BancRegistres;

architecture Behavioral of BancRegistres is
    --Creem unes senyals per seleccionar quina sortida del registre de 3 estats agafem
    signal select1 : STD_LOGIC_VECTOR (31 downto 0);
    signal select2 : STD_LOGIC_VECTOR (31 downto 0);
    signal Sortidreg : STD_LOGIC_VECTOR (31 downto 0);

    -- Cridem els components dels altres arxius
    component RegistreZero is
        Port (  clk : in STD_LOGIC;
                reset : in  STD_LOGIC;
                E1 : in STD_LOGIC;
                E2 : in STD_LOGIC;
                Dout1 : out STD_LOGIC_VECTOR(31 downto 0);
                Dout2 : out STD_LOGIC_VECTOR(31 downto 0));
    end component;

    component Descodificador5 is
    Port ( Habilitacio: in STD_LOGIC;
           Ent : in STD_LOGIC_VECTOR (4 downto 0);
           Sort : out STD_LOGIC_VECTOR (31 downto 0));
    end component;

    component RegSortida3Estats is
        Port ( Reset : in STD_LOGIC;
               clk : in STD_LOGIC;
               WE : in STD_LOGIC;
               Din : in STD_LOGIC_VECTOR (31 downto 0);
               E1 : in STD_LOGIC;
               E2 : in STD_LOGIC;
               Dout1 : out STD_LOGIC_VECTOR (31 downto 0);
               Dout2 : out STD_LOGIC_VECTOR (31 downto 0));
    end component;

begin
    -- Descodifiquem les entrades del banc per saber si tenim que escriure o no i quina sortida tenim que triar.
    dec1 : Descodificador5 port map(
           Habilitacio => lSor1,
           Ent => rf1,
           Sort => select1);
    dec2 : Descodificador5 port map(
           Habilitacio => lSor2,
           Ent => rf2,
           Sort => select2);
    dec3 : Descodificador5 port map(
           Habilitacio => esc,
           Ent => rdest,
           Sort => Sortidreg);
```

```
67      -- Creem els registres, el 0 és diferent als altres ja que només pot tenir el valor 0
68      reg0 : RegistreZero port map (clk,reset,select1(0),select2(0),sor1,sor2);
69      reg1 : RegSortida3Estats port map (reset,clk,Sortidreg(1),dades,select1(1),select2(1),sor1,sor2);
70      reg2 : RegSortida3Estats port map (reset,clk,Sortidreg(2),dades,select1(2),select2(2),sor1,sor2);
71      reg3 : RegSortida3Estats port map (reset,clk,Sortidreg(3),dades,select1(3),select2(3),sor1,sor2);
72      reg4 : RegSortida3Estats port map (reset,clk,Sortidreg(4),dades,select1(4),select2(4),sor1,sor2);
73      reg5 : RegSortida3Estats port map (reset,clk,Sortidreg(5),dades,select1(5),select2(5),sor1,sor2);
74      reg6 : RegSortida3Estats port map (reset,clk,Sortidreg(6),dades,select1(6),select2(6),sor1,sor2);
75      reg7 : RegSortida3Estats port map (reset,clk,Sortidreg(7),dades,select1(7),select2(7),sor1,sor2);
76      reg8 : RegSortida3Estats port map (reset,clk,Sortidreg(8),dades,select1(8),select2(8),sor1,sor2);
77      reg9 : RegSortida3Estats port map (reset,clk,Sortidreg(9),dades,select1(9),select2(9),sor1,sor2);
78      reg10 : RegSortida3Estats port map (reset,clk,Sortidreg(10),dades,select1(10),select2(10),sor1,sor2);
79      reg11 : RegSortida3Estats port map (reset,clk,Sortidreg(11),dades,select1(11),select2(11),sor1,sor2);
80      reg12 : RegSortida3Estats port map (reset,clk,Sortidreg(12),dades,select1(12),select2(12),sor1,sor2);
81      reg13 : RegSortida3Estats port map (reset,clk,Sortidreg(13),dades,select1(13),select2(13),sor1,sor2);
82      reg14 : RegSortida3Estats port map (reset,clk,Sortidreg(14),dades,select1(14),select2(14),sor1,sor2);
83      reg15 : RegSortida3Estats port map (reset,clk,Sortidreg(15),dades,select1(15),select2(15),sor1,sor2);
84      reg16 : RegSortida3Estats port map (reset,clk,Sortidreg(16),dades,select1(16),select2(16),sor1,sor2);
85      reg17 : RegSortida3Estats port map (reset,clk,Sortidreg(17),dades,select1(17),select2(17),sor1,sor2);
86      reg18 : RegSortida3Estats port map (reset,clk,Sortidreg(18),dades,select1(18),select2(18),sor1,sor2);
87      reg19 : RegSortida3Estats port map (reset,clk,Sortidreg(19),dades,select1(19),select2(19),sor1,sor2);
88      reg20 : RegSortida3Estats port map (reset,clk,Sortidreg(20),dades,select1(20),select2(20),sor1,sor2);
89      reg21 : RegSortida3Estats port map (reset,clk,Sortidreg(21),dades,select1(21),select2(21),sor1,sor2);
90      reg22 : RegSortida3Estats port map (reset,clk,Sortidreg(22),dades,select1(22),select2(22),sor1,sor2);
91      reg23 : RegSortida3Estats port map (reset,clk,Sortidreg(23),dades,select1(23),select2(23),sor1,sor2);
92      reg24 : RegSortida3Estats port map (reset,clk,Sortidreg(24),dades,select1(24),select2(24),sor1,sor2);
93      reg25 : RegSortida3Estats port map (reset,clk,Sortidreg(25),dades,select1(25),select2(25),sor1,sor2);
94      reg26 : RegSortida3Estats port map (reset,clk,Sortidreg(26),dades,select1(26),select2(26),sor1,sor2);
95      reg27 : RegSortida3Estats port map (reset,clk,Sortidreg(27),dades,select1(27),select2(27),sor1,sor2);
96      reg28 : RegSortida3Estats port map (reset,clk,Sortidreg(28),dades,select1(28),select2(28),sor1,sor2);
97      reg29 : RegSortida3Estats port map (reset,clk,Sortidreg(29),dades,select1(29),select2(29),sor1,sor2);
98      reg30 : RegSortida3Estats port map (reset,clk,Sortidreg(30),dades,select1(30),select2(30),sor1,sor2);
99      reg31 : RegSortida3Estats port map (reset,clk,Sortidreg(31),dades,select1(31),select2(31),sor1,sor2);
100   end Behavioral;
```

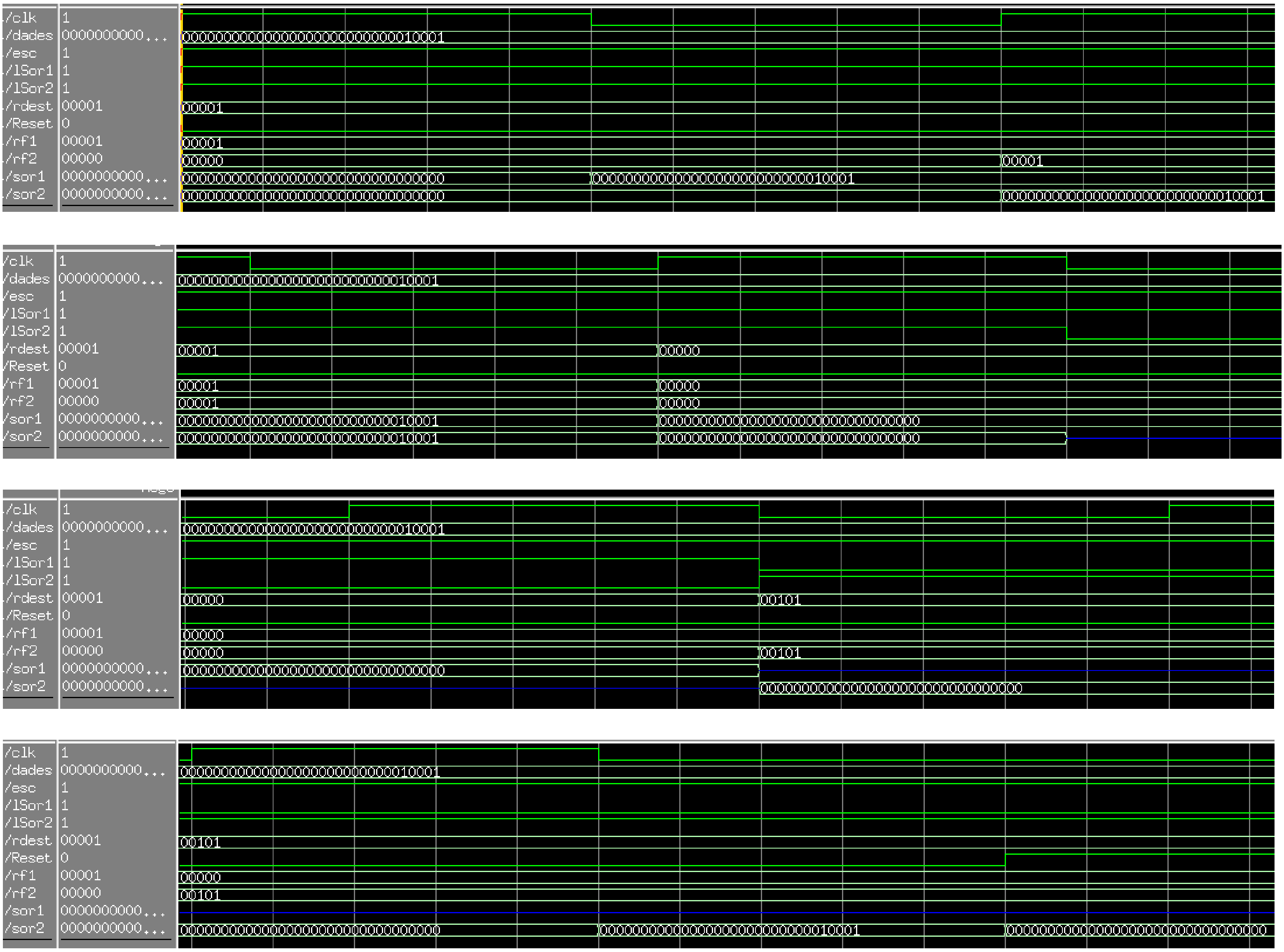
Un cop hem creat els 32 registres creem un test .

TEST BENCH BANC DE REGISTRES

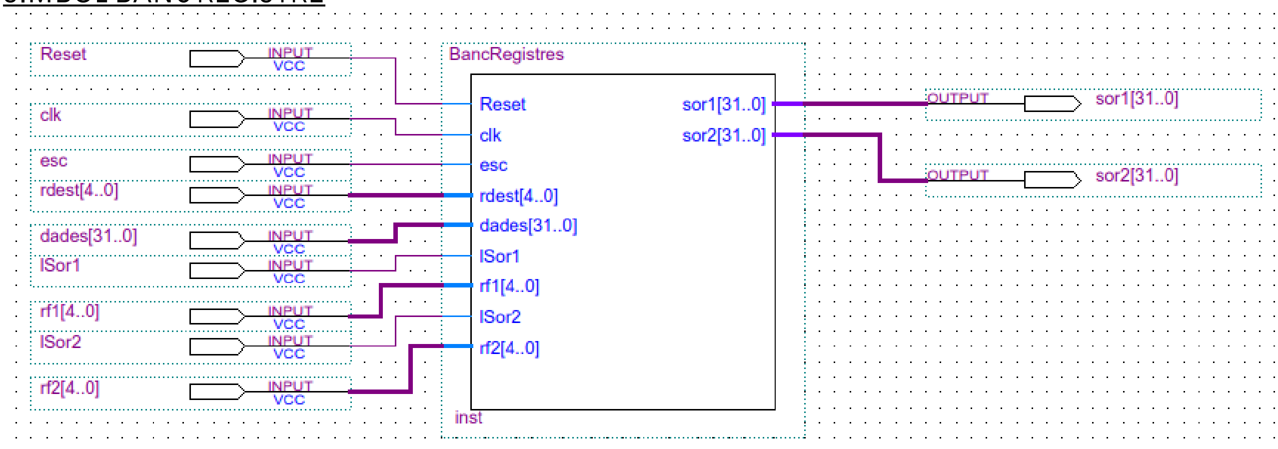
```
77    init : PROCESS
78    -- variable declarations
79    BEGIN
80       clk <= '1';
81       dades <=x"00000011";
82       esc <= '1';
83       lSor1 <= '1';
84       lSor2 <= '1';
85       rdest <= "00001";
86       Reset <= '0';
87       rf1 <= "00001";
88       rf2 <= "00000";
89       wait for 10 us;
90       clk <= '0';
91       wait for 10 us;
92       clk <= '1';
93       rdest <= "00001";
94       rf1 <= "00001";
95       rf2 <= "00001";
96       wait for 10 us;
97       clk <= '0';
98       wait for 10 us;
99       clk <= '1';
100      rdest <= "00000";
```

```
101      rf1 <= "00000";
102      rf2 <= "00000";
103      wait for 10 us;
104      clk <= '0';
105      lSor1 <= '1';
106      lSor2 <= '0';
107      wait for 10 us;
108      clk <= '1';
109      wait for 10 us;
110      clk <= '0';
111      lSor1 <= '0';
112      lSor2 <= '1';
113      rdest <= "00101";
114      rf1 <= "00000";
115      rf2 <= "00101";
116      wait for 10 us;
117      clk <='1';
118      wait for 10 us;
119      clk <='0';
120      wait for 10 us;
121      reset <= '1';
122      wait for 10 us;
123   WAIT;
```

GRAFICA BANC DE REGISTRES

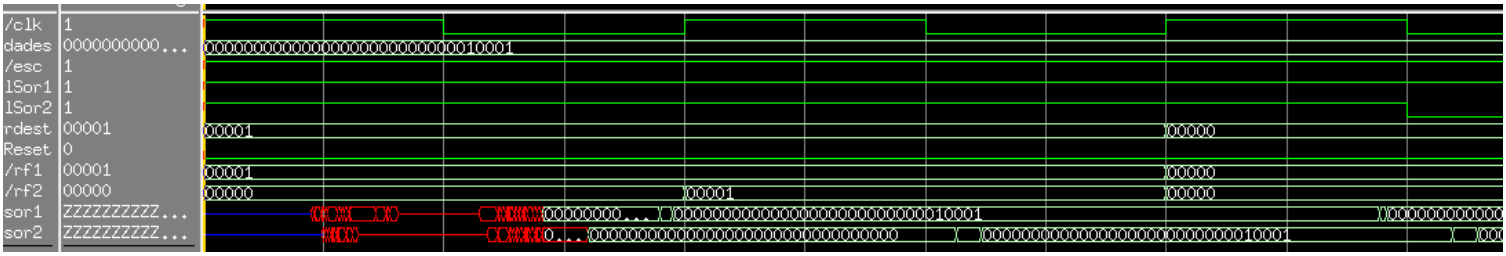


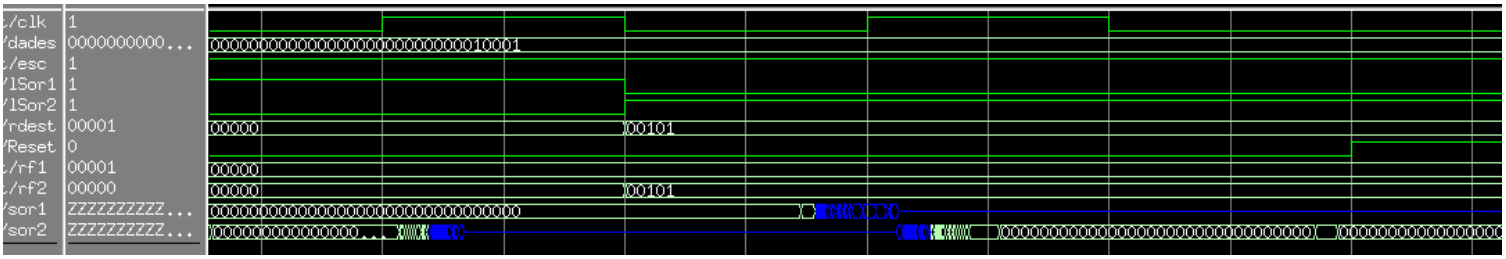
SIMBOL BANC REGISTRE



Finalment simulem en banc de registres a nivell de porta lògica, o sigui en cas real.

SIMULACIÓ GRÀFICA REAL BANC DE REGISTRES





Podem veure com es produeixen retards i com les portes lògiques introdueixen alguns problemes. Això és degut al baix temps que hem ficat en el test bench entre clock i clock.

Finalment dissenyem el PC (Program Counter), que ens permetrà incrementar en quatre unitats el seu contingut a cada cicle d'instrucció i actualitzar el seu valor amb una dada nova en el cas de les instruccions de salt.

Per tant, en el flanc de pujada del rellotge es captura el valor actual del PC que guardarem en la senyal LecturaPC, que sera la que posarem al bus quan ho ordeni un senyal de lectura. En canvi, en el flanc de baixada es quan realitzarem les funcions d'actualització i modificació del PC utilitzant la senyal ContingutPC.

PROGRAM COUNTER

```
1   library ieee;                    --Cridem les llibreries necessaries.
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_unsigned.all;
4   entity ComptadorPrograma is
5      Port ( Reset : in STD_LOGIC;  --Inicialitza PC=0.
6             clk : in STD_LOGIC;
7             IncPC : in STD_LOGIC;   --Increment del PC+4.
8             LPC : in STD_LOGIC;     --Lectura del valor PC.
9             EPC : in STD_LOGIC;     --Actualitza valor PC.
10            Din : in STD_LOGIC_VECTOR (31 downto 0);  --Valor per realitzar salt en PC.
11            Dout : out STD_LOGIC_VECTOR (31 downto 0)); --Valor de sortida del PC.
12  end ComptadorPrograma;
13  architecture PC_Arch of ComptadorPrograma is
14  --Creem dos senyals per llegir i guardar el contingut del PC.
15  signal ContingutPC: STD_LOGIC_VECTOR (31 downto 0):= x"00000000";
16  signal LecturaPC: STD_LOGIC_VECTOR (31 downto 0):= x"00000000";
17  begin
18  process (clk, reset)
19  begin
20     if rising_edge(clk) then
21        if reset = '1' then
22           LecturaPC <= x"00000000";
23        else LecturaPC <= ContingutPC;
24        end if;
25     end if;
26  end process;
```

```
27  process (clk,reset)
28  begin
29     if (falling_edge(clk)) then
30        if reset = '1' then                    --Posem PC=0.
31           ContingutPC <= x"00000000";
32        elsif IncPC='1' then                   --Incrementem PC=PC+4.
33           ContingutPC <= ContingutPC + 4;
34        elsif EPC='1' then
35           ContingutPC <= Din;                 --Actualitzem PC=Din.
36        end if;
37     end if;
38  end process;
39  Dout <= LecturaPC when LPC = '1' else "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ";
40  end PC_Arch;
```

Un cop tenim el disseny del PC acabat realitzem una prova del circuit aplicant uns valors determinats per poder comprovar el seu correcte funcionament.

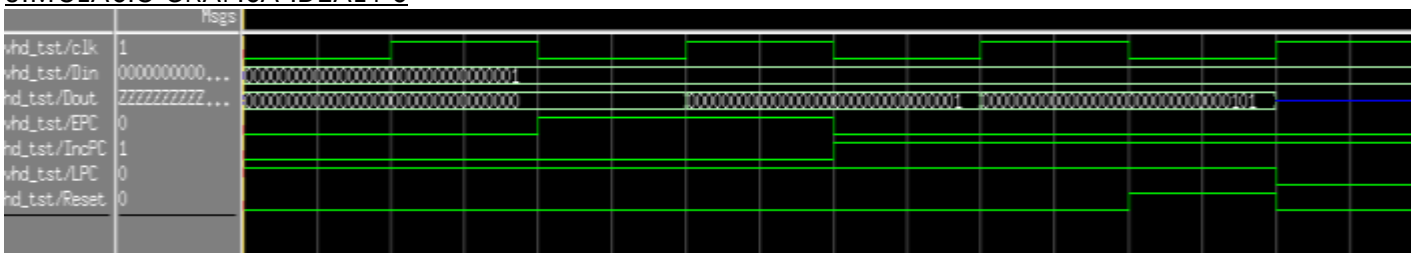
TEST BENCH PC

```
27    LIBRARY ieee;
28    USE ieee.std_logic_1164.all;
29
30    ENTITY ComptadorPrograma_vhd_tst IS
31    END ComptadorPrograma_vhd_tst;
32    ARCHITECTURE ComptadorPrograma_arch OF ComptadorPrograma_vhd_tst IS
33    -- constants
34    -- signals
35    SIGNAL clk : STD_LOGIC;
36    SIGNAL Din : STD_LOGIC_VECTOR(31 DOWNTO 0);
37    SIGNAL Dout : STD_LOGIC_VECTOR(31 DOWNTO 0);
38    SIGNAL EPC : STD_LOGIC;
39    SIGNAL IncPC : STD_LOGIC;
40    SIGNAL LPC : STD_LOGIC;
41    SIGNAL Reset : STD_LOGIC;
42    COMPONENT ComptadorPrograma
43       PORT (
44       clk : IN STD_LOGIC;
45       Din : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
46       Dout : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
47       EPC : IN STD_LOGIC;
48       IncPC : IN STD_LOGIC;
49       LPC : IN STD_LOGIC;
50       Reset : IN STD_LOGIC
51       );
52    END COMPONENT;
53    BEGIN
54       i1 : ComptadorPrograma
55       PORT MAP (
56    -- list connections between master ports and signals
57       clk => clk,
58       Din => Din,
59       Dout => Dout,
60       EPC => EPC,
61       IncPC => IncPC,
62       LPC => LPC,
63       Reset => Reset
64       );
```

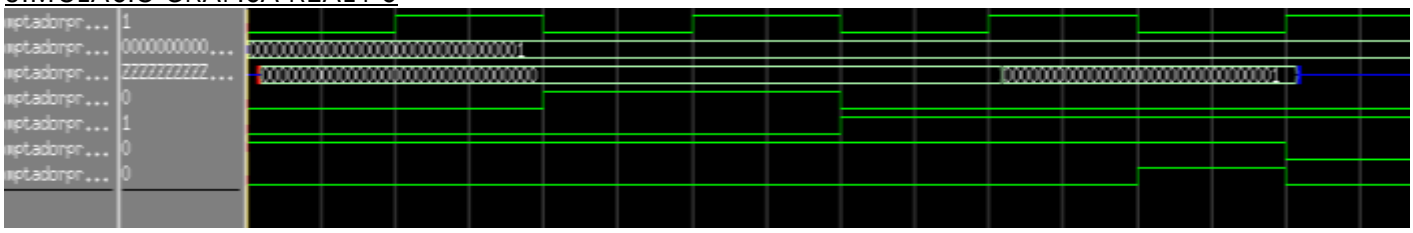
```
65  init : PROCESS
66  -- variable declarations
67  BEGIN
68      clk <= '0';
69      Din <= x"00000001";
70      EPC <= '0';
71      IncPC <= '0';
72      LPC <= '1';
73      Reset <='0';
74      wait for 10 ns;
75      clk <= '1';
76      wait for 10 ns;
77      clk <= '0';
78      EPC <= '1';
79      --IncPC <= '1';
80      wait for 10 ns;
81      clk <= '1';
82      wait for 10 ns;
83      clk <= '0';
84      EPC <= '0';
85      IncPC <= '1';
86      wait for 10 ns;
87      clk <= '1';
88      wait for 10 ns;
89      clk <= '0';
90      reset <= '1';
91      wait for 10 ns;
92      clk <= '1';
93      reset <= '0';
94      LPC <= '0';
95      wait for 10 ns;
96
97  WAIT;
98  END PROCESS init;
99  always : PROCESS
100 BEGIN
101 WAIT;
102 END PROCESS always;
103 END ComptadorPrograma_arch;
```

Un cop tenim el programa de prova creem les gràfiques que ens mostrant els resultats visualment observant la diferencia entre elles si es ten en compte els retards.

SIMULACIÓ GRÀFICA IDEAL PC



SIMULACIÓ GRÀFICA REAL PC



Per ultim realitzarem el símbol del PC utilitzant el disseny anterior.

SIMBOL PC