Pràctica 2:

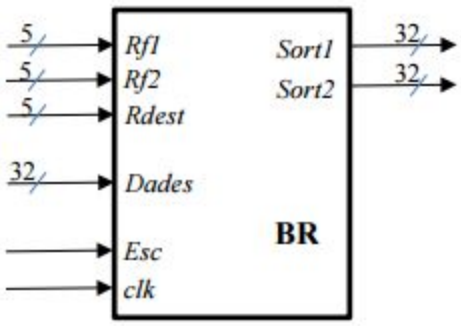
Disseny del banc de registres i del comptador de programa

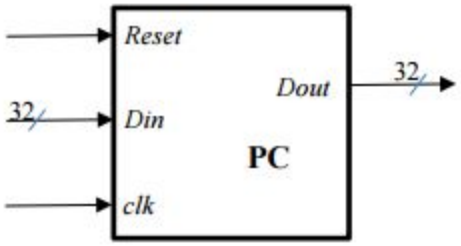
Arnau Plans Castelló
Marc Estiarte Salisi
Eloi Galan Badiella

Índex

1. Introducció

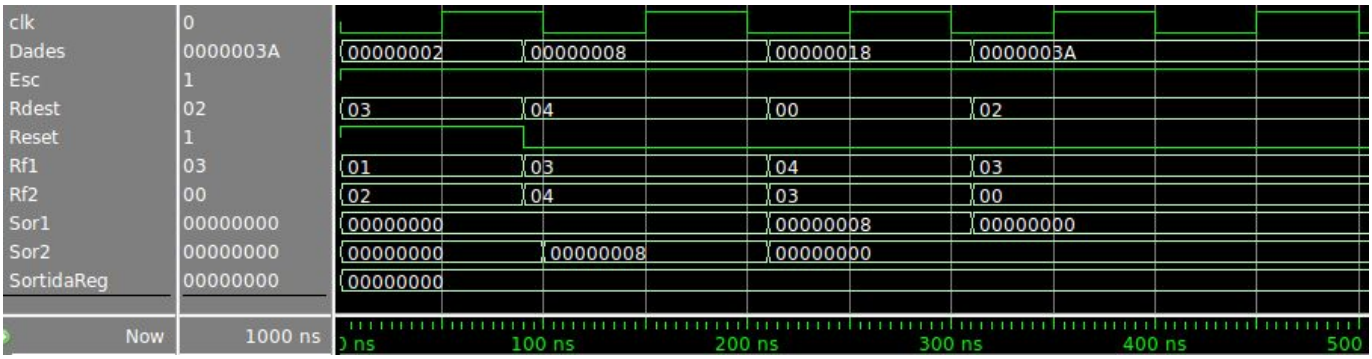
En aquesta pràctica realitzarem el disseny del Banc de Registres i del Comptador de Programa que hem vist a teoria.
Aquest disseny permetrà executar un conjunt reduït d'instruccions relacionades amb la unitat aritmètica i lògica.

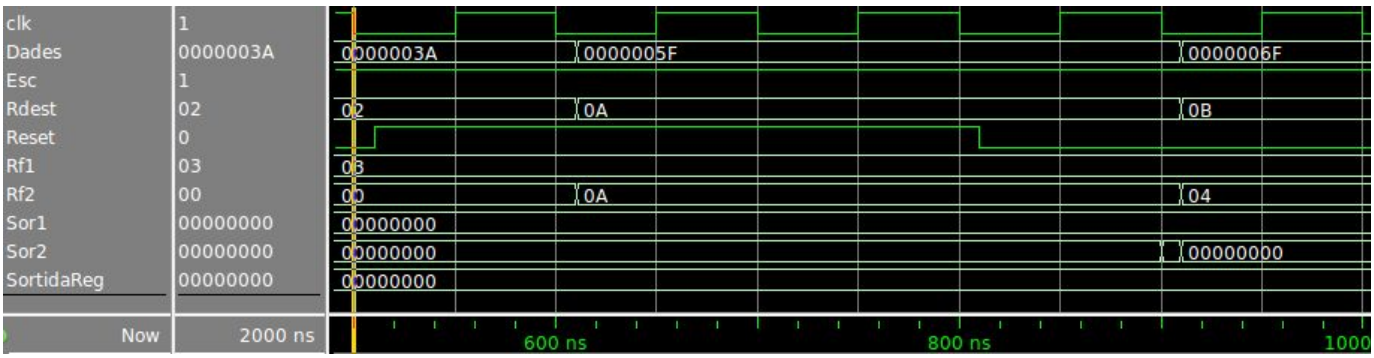




2. Disseny BancRegistres i PC

Simulació (Banc de Registres)





Explicació de la simulació del Banc de Registres:

En la primera captura de la simulació del Banc de Registres podem observar el següent:

0 ns - 100 ns → Activem el reset per tal d'assegurar-nos que no hi ha cap valor no desitjat a cap registre. Podem observar que al Rdest 3 no se l'hi emmagatzema cap dada ja que el reset està activat. Quan queda poc temps per acabar el primer període del clock desactivem el reset, canviem el Rdest a 4, Rf1 a 3 i Rf2 a 4.

100 ns - 200 ns → Podem veure com canvia el valor de la sortida del registre 4 al qual l'hi hem assignat el valor 8.

200 ns - 300 ns → En aquest punt hem volgut assignar un valor diferent a 0 al registre0 per veure el seu correcte funcionament.

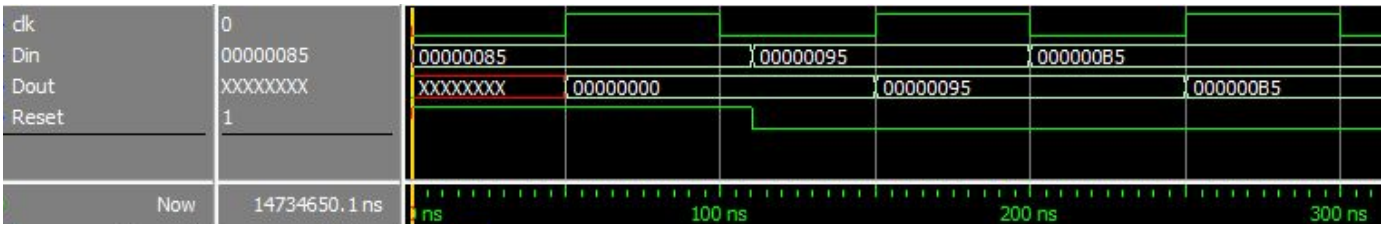
300 ns - 400 ns → Tal i com esperàvem, la sortida del registre0 després d'assignar-li un valor ha estat 0. (ja que el valor del registre0 no pot ser mai diferent de 0).

400 ns - 900 ns → En aquest punt hem volgut activar el reset per comprobar si tots els registres es tornaven a reinicialitzar.

900 ns - 1000 ns → Com era d'esperar, hem observat el registre 3 i 4 que anteriorment els havíem assignat valors, i el seu valor actual tornava a ser 0. Per tant podem afirmar que el reset funciona perfectament.

En general podem observar com la escriptura dels registres es Sincrona i la lectura dels registres és Asincrona.

Simulació (PC)

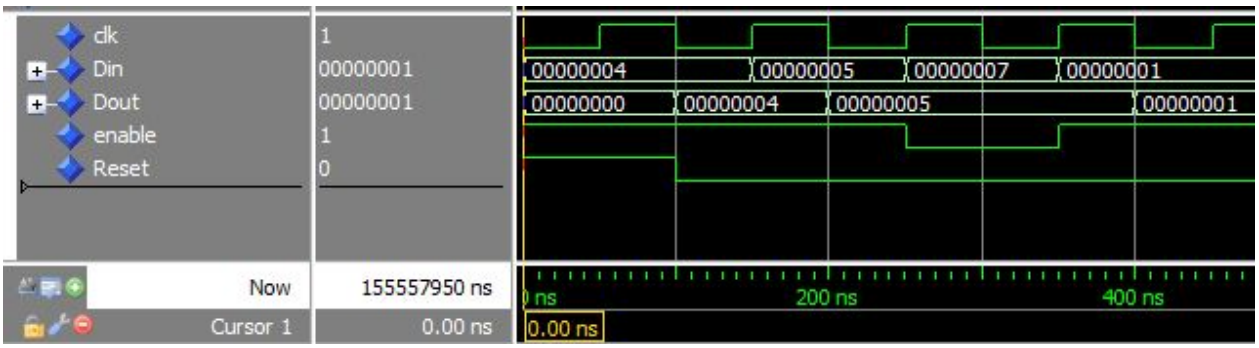


Explicació de la simulació del PC:

En aquesta simulació hem comprovat que l'entrada inserida com a Din al Comptador de programa ha de ser la sortida que surt com a Dout d'aquest mateix de manera síncrona. En aquest model de comptador de programa no hem utilitzat cap sumador intern ja que la suma d'aquest mateix es farà en una altre unitat purament enfocada a la suma.

0 ns - 50 ns → Al comprovar el senyal de reset al flanc de pujada l'estat de la sortida és indefinit ja que fins al "primer" flanc de pujada no sabem com està el reset i per tant no podem saber el output del senyal.

Simulació (Registre)

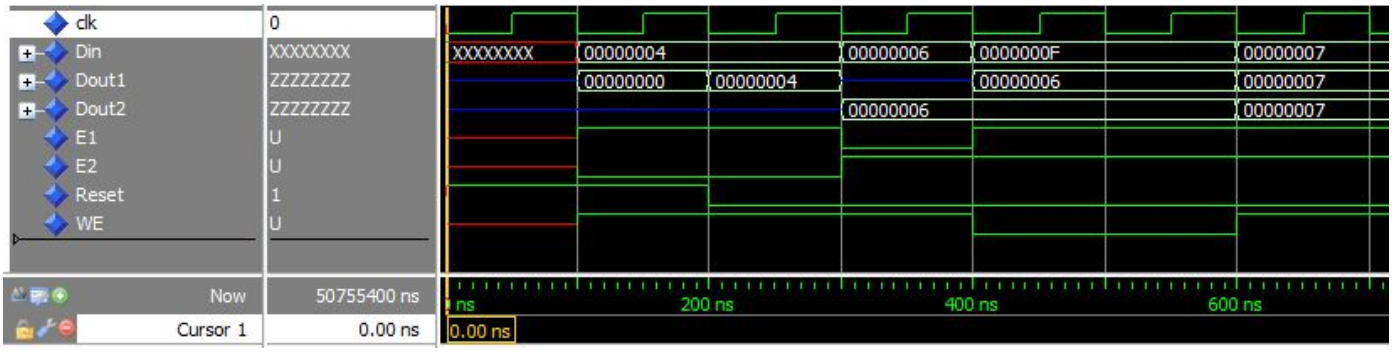


Explicació de la simulació del Registre:

En aquesta simulació com podem veure, a no ser que el Reset es trobi actiu a 1, el qual significa que la sortida serà tot 0, la resta de proves en la sortida Dout tenim el mateix que tindria al input, ja que s'ha executat una escriptura en el registre.

S'ha de destacar que la escriptura és sincronia i que per tant segueix el clock en la llista de sensibilitats, ara bé, aquesta sincronia es te en compte en el flanc de baixada, per tant només s'escriurà en el flanc de baixada.

Simulació (Registre Sortida 3 estats)



Explicació de la simulació del Registre Sortida 3 estats:

En la simulació del registre de tres estats hem comprovat com depenent del enable que habilitis aconsegueixes escriure en una sortida o en una altre, pel cas de activar E1 com podem veure, la sortida Dout1 s'activa, i pel contrari si desactivem E1 i activem E2, Dout1 no tenim cap output però per Dout2 tenim l'output esperat.

Com a última prova hem provat d'activar els 2 enables a la vegada E1 = '1' i E2 = '2', i com a resultat hem tingut els 2 outputs per cada sortida en els seus moments pertinents.

3. Codi complet

3.1. VHD

Banc Resgistres:

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

entity BancRegistres is
   port (
      Rf1: STD_LOGIC_VECTOR(4 downto 0); -- registres font
      Rf2: STD_LOGIC_VECTOR(4 downto 0);
      Rdest: STD_LOGIC_VECTOR(4 downto 0); -- registre destí

      Sor1 : out STD_LOGIC_VECTOR(31 downto 0); -- registres sortida
      Sor2 : out STD_LOGIC_VECTOR(31 downto 0);

      Dades: in STD_LOGIC_VECTOR(31 downto 0); -- registre de Dades
      -- indica entrada informació a escriure sobre Rdest

      Esc : in STD_LOGIC; -- si 1 llegim Dades i escrivim a rdest
      -- captura la dada de Dades i emmagatzema en Rdest
                  Reset : in STD_LOGIC;

      clk : in STD_LOGIC -- sincronitza escriptura en registres
                  );
end BancRegistres;


architecture Behavioral of BancRegistres is -- comportament del B.R.
   component RegSortida3Estats is
      Port (
         E1    : in STD_LOGIC; -- identifiquen bus sortida
         E2    : in STD_LOGIC;
         Din   : in STD_LOGIC_VECTOR (31 downto 0); -- ídem Dades

         Dout1 : out STD_LOGIC_VECTOR (31 downto 0); -- camí de Dades
         Dout2 : out STD_LOGIC_VECTOR (31 downto 0);

         WE    : in STD_LOGIC; -- habilita escriptura en algun dels registres

         Reset : in STD_LOGIC;
```

```
        clk   : in STD_LOGIC -- definit en flanc de baixada
    );
  end component;

  component RegistreZero is
    Port (
        E1    : in STD_LOGIC; -- identifiquen bus sortida
        E2    : in STD_LOGIC;
        Dout1 : out STD_LOGIC_VECTOR (31 downto 0); -- camí de Dades
        Dout2 : out STD_LOGIC_VECTOR (31 downto 0);
        Reset : in STD_LOGIC;
        clk   : in STD_LOGIC -- definit en flanc de baixada
    );
  end component;
  -- si es fa el registre de forma comportamental dins de la definició d'arquitectura no farà falta la definició del
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;
  signal RegSel1: STD_LOGIC_VECTOR (31 downto 0);
  signal RegSel2: STD_LOGIC_VECTOR (31 downto 0);
  signal RegEsc: STD_LOGIC_VECTOR(31 downto 0);
  begin
  dec1: Descodificador5 port map (Habilitacio => '1', Ent => Rf1, Sort => RegSel1);
  dec2: Descodificador5 port map (Habilitacio => '1', Ent => Rf2, Sort => RegSel2);
  dec3: Descodificador5 port map (Habilitacio => esc, Ent => Rdest, Sort => RegEsc);

        reg0: RegistreZero port map (Reset => Reset, clk =>clk, E1 => RegSel1(0), E2 => RegSel2(0), Dout1 =>
Sor1, Dout2 => Sor2);
        reg1: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(1), Din => Dades, E1 =>
RegSel1(1), E2 => RegSel2(1), Dout1 => Sor1, Dout2 => Sor2);
        reg2: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(2), Din => Dades, E1 =>
RegSel1(2), E2 => RegSel2(2), Dout1 => Sor1, Dout2 => Sor2);
        reg3: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(3), Din => Dades, E1 =>
RegSel1(3), E2 => RegSel2(3), Dout1 => Sor1, Dout2 => Sor2);
        reg4: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(4), Din => Dades, E1 =>
RegSel1(4), E2 => RegSel2(4), Dout1 => Sor1, Dout2 => Sor2);
        reg5: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(5), Din => Dades, E1 =>
RegSel1(5), E2 => RegSel2(5), Dout1 => Sor1, Dout2 => Sor2);
        reg6: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(6), Din => Dades, E1 =>
RegSel1(6), E2 => RegSel2(6), Dout1 => Sor1, Dout2 => Sor2);
        reg7: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(7), Din => Dades, E1 =>
RegSel1(7), E2 => RegSel2(7), Dout1 => Sor1, Dout2 => Sor2);
        reg8: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(8), Din => Dades, E1 =>
RegSel1(8), E2 => RegSel2(8), Dout1 => Sor1, Dout2 => Sor2);
```

```
        reg9: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(9), Din => Dades, E1 =>
RegSel1(9), E2 => RegSel2(9), Dout1 => Sor1, Dout2 => Sor2);
        reg10: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(10), Din => Dades, E1 =>
RegSel1(10), E2 => RegSel2(10), Dout1 => Sor1, Dout2 => Sor2);
        reg11: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(11), Din => Dades, E1 =>
RegSel1(11), E2 => RegSel2(11), Dout1 => Sor1, Dout2 => Sor2);
        reg12: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(12), Din => Dades, E1 =>
RegSel1(12), E2 => RegSel2(12), Dout1 => Sor1, Dout2 => Sor2);
        reg13: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(13), Din => Dades, E1 =>
RegSel1(13), E2 => RegSel2(13), Dout1 => Sor1, Dout2 => Sor2);
        reg14: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(14), Din => Dades, E1 =>
RegSel1(14), E2 => RegSel2(14), Dout1 => Sor1, Dout2 => Sor2);
        reg15: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(15), Din => Dades, E1 =>
RegSel1(15), E2 => RegSel2(15), Dout1 => Sor1, Dout2 => Sor2);
        reg16: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(16), Din => Dades, E1 =>
RegSel1(16), E2 => RegSel2(16), Dout1 => Sor1, Dout2 => Sor2);
        reg17: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(17), Din => Dades, E1 =>
RegSel1(17), E2 => RegSel2(17), Dout1 => Sor1, Dout2 => Sor2);
        reg18: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(18), Din => Dades, E1 =>
RegSel1(18), E2 => RegSel2(18), Dout1 => Sor1, Dout2 => Sor2);
        reg19: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(19), Din => Dades, E1 =>
RegSel1(19), E2 => RegSel2(19), Dout1 => Sor1, Dout2 => Sor2);
        reg20: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(20), Din => Dades, E1 =>
RegSel1(20), E2 => RegSel2(20), Dout1 => Sor1, Dout2 => Sor2);
        reg21: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(21), Din => Dades, E1 =>
RegSel1(21), E2 => RegSel2(21), Dout1 => Sor1, Dout2 => Sor2);
        reg22: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(22), Din => Dades, E1 =>
RegSel1(22), E2 => RegSel2(22), Dout1 => Sor1, Dout2 => Sor2);
        reg23: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(23), Din => Dades, E1 =>
RegSel1(23), E2 => RegSel2(23), Dout1 => Sor1, Dout2 => Sor2);
        reg24: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(24), Din => Dades, E1 =>
RegSel1(24), E2 => RegSel2(24), Dout1 => Sor1, Dout2 => Sor2);
        reg25: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(25), Din => Dades, E1 =>
RegSel1(25), E2 => RegSel2(25), Dout1 => Sor1, Dout2 => Sor2);
        reg26: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(26), Din => Dades, E1 =>
RegSel1(26), E2 => RegSel2(26), Dout1 => Sor1, Dout2 => Sor2);
        reg27: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(27), Din => Dades, E1 =>
RegSel1(27), E2 => RegSel2(27), Dout1 => Sor1, Dout2 => Sor2);
        reg28: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(28), Din => Dades, E1 =>
RegSel1(28), E2 => RegSel2(28), Dout1 => Sor1, Dout2 => Sor2);
        reg29: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(29), Din => Dades, E1 =>
RegSel1(29), E2 => RegSel2(29), Dout1 => Sor1, Dout2 => Sor2);
        reg30: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(30), Din => Dades, E1 =>
RegSel1(30), E2 => RegSel2(30), Dout1 => Sor1, Dout2 => Sor2);
        reg31: RegSortida3Estats port map (Reset => Reset, clk =>clk, WE=>RegEsc(31), Din => Dades, E1 =>
RegSel1(31), E2 => RegSel2(31), Dout1 => Sor1, Dout2 => Sor2);
end architecture;
```

Comptador programa:

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

entity ComptadorPrograma is
	Port(Reset : in STD_LOGIC;
		clk : in STD_LOGIC;
		Din : in STD_LOGIC_VECTOR(31 downto 0);
		Dout : out STD_LOGIC_VECTOR(31 downto 0));
end ComptadorPrograma;

architecture ArqPC of ComptadorPrograma is
	signal ContingutPC : STD_LOGIC_VECTOR(31 downto 0);
	begin
		process(clk,reset)
		begin
			if rising_edge (clk) then
				if(reset = '1') then
					ContingutPC <= (others => '0');
				else
					ContingutPC <= Din;
				end if;
			end if;
		end process;
	Dout <=  ContingutPC;
end ArqPC;
```

Descodificador5:

```
library ieee;
use ieee.std_logic_1164.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 Behavioral of Descodificador5 is
   signal Sortida: STD_LOGIC_VECTOR(31 downto 0);
   begin
      with Ent select
```

```
    Sortida <=   "00000000000000000000000000000001" when "00000",
                 "00000000000000000000000000000010" when "00001",
                 "00000000000000000000000000000100" when "00010",
                 "00000000000000000000000000001000" when "00011",
                 "00000000000000000000000000010000" when "00100",
                 "00000000000000000000000000100000" when "00101",
                 "00000000000000000000000001000000" when "00110",
                 "00000000000000000000000010000000" when "00111",
                 "00000000000000000000000100000000" when "01000",
                 "00000000000000000000001000000000" when "01001",
                 "00000000000000000000010000000000" when "01010",
                 "00000000000000000000100000000000" when "01011",
                 "00000000000000000001000000000000" when "01100",
                 "00000000000000000010000000000000" when "01101",
                 "00000000000000000100000000000000" when "01110",
                 "00000000000000001000000000000000" when "01111",
                 "00000000000000010000000000000000" when "10000",
                 "00000000000000100000000000000000" when "10001",
                 "00000000000001000000000000000000" when "10010",
                 "00000000000010000000000000000000" when "10011",
                 "00000000000100000000000000000000" when "10100",
                 "00000000001000000000000000000000" when "10101",
                 "00000000010000000000000000000000" when "10110",
                 "00000000100000000000000000000000" when "10111",
                 "00000001000000000000000000000000" when "11000",
                 "00000010000000000000000000000000" when "11001",
                 "00000100000000000000000000000000" when "11010",
                 "00001000000000000000000000000000" when "11011",
                 "00010000000000000000000000000000" when "11100",
                 "00100000000000000000000000000000" when "11101",
                 "01000000000000000000000000000000" when "11110",
                 "10000000000000000000000000000000" when "11111",
                 "00000000000000000000000000000000" when others;

   Sort <= Sortida when Habilitacio = '1' else (others => 'Z');

end architecture;
```

Registre:

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

entity Registre is
   port (
      Reset  : in STD_LOGIC;
      clk    : in STD_LOGIC;
      enable : in STD_LOGIC;
```

```
      Din    : in STD_LOGIC_VECTOR(31 downto 0);
      Dout   : out STD_LOGIC_VECTOR(31 downto 0)
   );
end Registre;

architecture Behavioral of Registre is
begin
   -- Es pot realitzar una definició comportamental del registre, basat en un procés
process(clk,Reset)
begin
   if(Reset = '1') then
      Dout <= (others => '0');
   elsif falling_edge(clk) and (enable = '1') then
      Dout <= Din;
   end if;
end process;
end Behavioral;
```

Registre Zero:

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

entity RegistreZero is
        port( Reset: in STD_LOGIC;
                        clk: 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 RegistreZero;

architecture Behavioral of RegistreZero is
begin
        Dout1 <= (others => '0') when E1 = '1' else (others => 'Z');
        Dout2 <= (others => '0') when E2 = '1' else (others => 'Z');
end Behavioral;
```

Registre sortida 3 estats:

```
library ieee;
use ieee.std_logic_1164.all;
entity 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 RegSortida3Estats;
architecture Behavioral of RegSortida3Estats is
    component Registre is
        port(
            Reset  : in STD_LOGIC;
            clk    : in STD_LOGIC;
            Enable : in STD_LOGIC;
            Din    : in STD_LOGIC_VECTOR(31 downto 0);
            Dout   : out STD_LOGIC_VECTOR(31 downto 0)
        );
    end component;

    signal SortidaReg: STD_LOGIC_VECTOR(31 downto 0);
    begin
    reg: Registre port map (
        Reset  => Reset,
        clk    => clk,
        Enable => WE,
        Din    => Din,
        Dout   => SortidaReg);

        Dout1  <= SortidaReg when E1 = '1' else (others => 'Z');
        Dout2  <= SortidaReg when E2 = '1' else (others => 'Z');
end Behavioral;
```

3.2. VHT (TestBench)

Clock de 100ns de periode.

Banc de registres:

```
Reset <= '1'; --Activem reset

Rf1 <= "00001"; --r1
Rf2 <= "00010"; --r2
Rdest <= "00011"; --r3
Dades <= "00000000000000000000000000000010";
Esc <= '1';

wait for 90 ns;
Reset <= '0'; --Desactivem el reset

Rf1 <= "00011"; --r3
Rf2 <= "00100"; --r4
Rdest <= "00100"; --r4
Dades <= "00000000000000000000000000001000";
Esc <= '1';
wait for 120 ns;

Rf1 <= "00100"; --r4
Rf2 <= "00011"; --r3
Rdest <= "00000"; --r0
Dades <= "00000000000000000000000000011000";
Esc <= '1';
wait for 100 ns;

Rf1 <= "00011"; --r3
Rf2 <= "00000"; --r0
Rdest <= "00010"; --r2
Dades <= "00000000000000000000000000111010";
Esc <= '1';
wait for 100 ns;


wait for 100 ns;
Reset <= '1'; --Activem el reset
wait for 100 ns;

Rf1 <= "00011"; --r4
```

```
Rf2 <= "01010"; --r10
Rdest <= "01010"; --r4
Dades <= "00000000000000000000000001011111";
Esc <= '1';
wait for 100 ns;


wait for 100 ns;
Reset <= '0'; --Desactivem el reset
wait for 100 ns;

Rf1 <= "00011"; --r3
Rf2 <= "00100"; --r4
Rdest <= "01011"; --r1
Dades <= "00000000000000000000000001101111";
Esc <= '1';
wait for 130 ns;
```

Registre Sortida 3 estats:

```
Reset <= '1';
wait for 100 ns;

WE    <= '1';
Din   <= "00000000000000000000000000000100";
E1    <= '1';
E2          <= '0';

wait for 100 ns;
Reset <= '0';
wait for 100 ns;

WE    <= '1';
Din   <= "00000000000000000000000000000110";
E1    <= '0';
E2          <= '1';

wait for 100 ns;
WE    <= '0';
Din   <= "00000000000000000000000000001111";
E1    <= '1';
E2          <= '1';

wait for 100 ns;

wait for 100 ns;
```

```
WE    <= '1';
Din   <= "00000000000000000000000000000111";
E1    <= '1';
E2          <= '1';
```

Registre:

```
Reset  <= '1';
enable <= '1';
Din    <= "00000000000000000000000000000100";
wait for 100ns;

Reset  <= '0';
wait for 50ns;

enable <= '1';
Din    <= "00000000000000000000000000000101";
wait for 100ns;

enable <= '0';
Din    <= "00000000000000000000000000000111";
wait for 100ns;

enable <= '1';
Din    <= "00000000000000000000000000000001";
wait for 100ns;
```

Comptador de Programa:

```
Reset <= '1';
Din <= "00000000000000000000000010000101";  -- 0x85
wait for 110 ns;

Reset <= '0';
Din <= "00000000000000000000000010010101"; -- 0x95
wait for 90 ns;

Reset <= '0';
Din <= "00000000000000000000000010110101"; -- 0xB5
wait for 90 ns;
```