Gestion de parking

Réalisé par:
Ghazouani Chaima

A. L'architecture du système

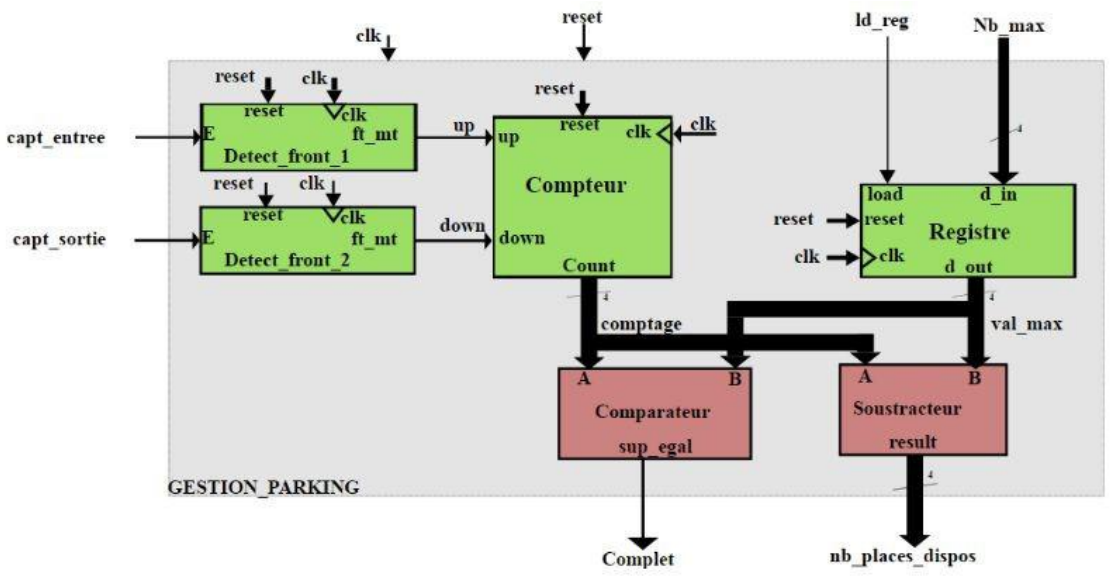
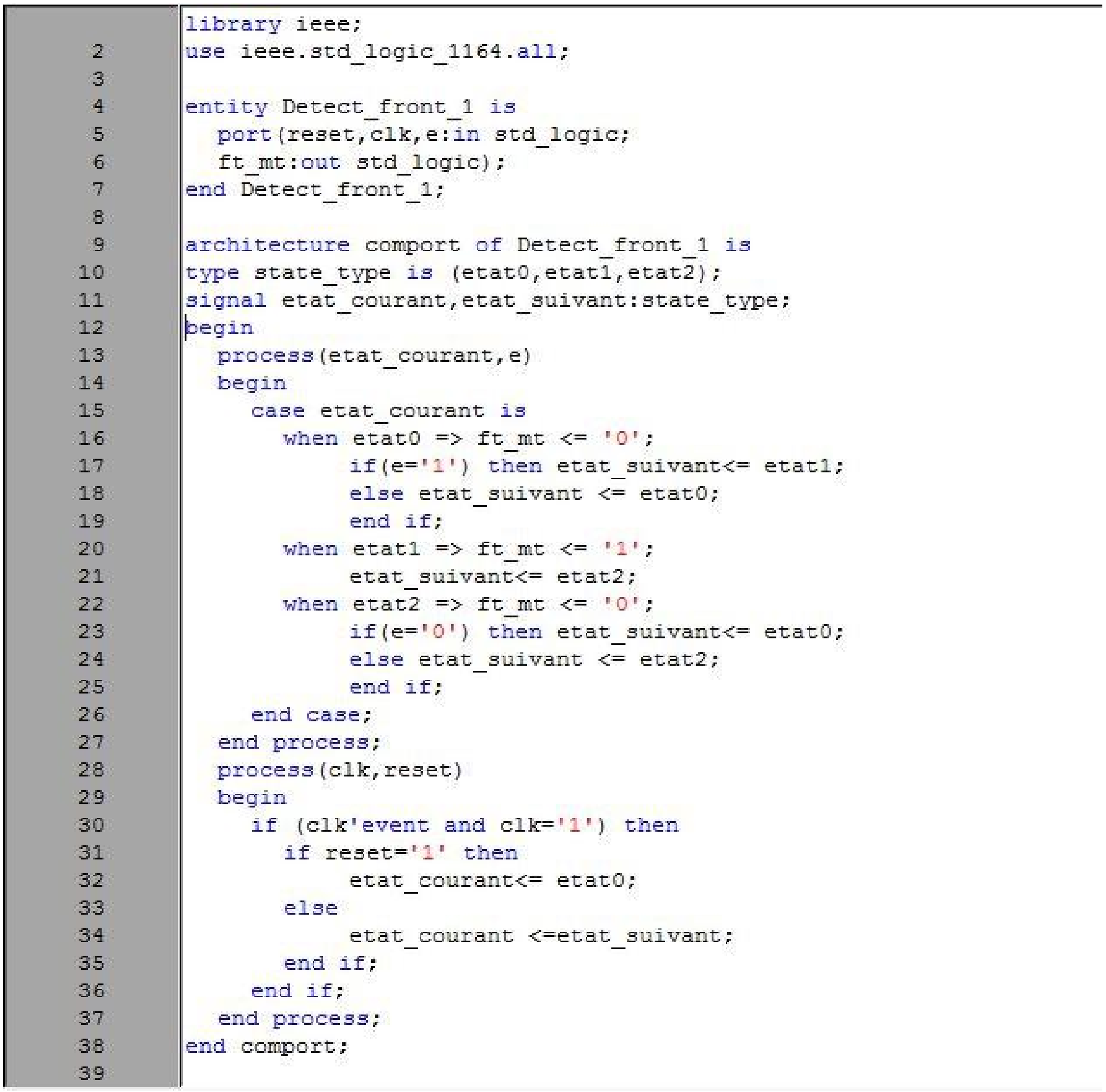


Figure 1 : architecture du système de gestion du parking

B. Programmation et simulation des différentes parties du système

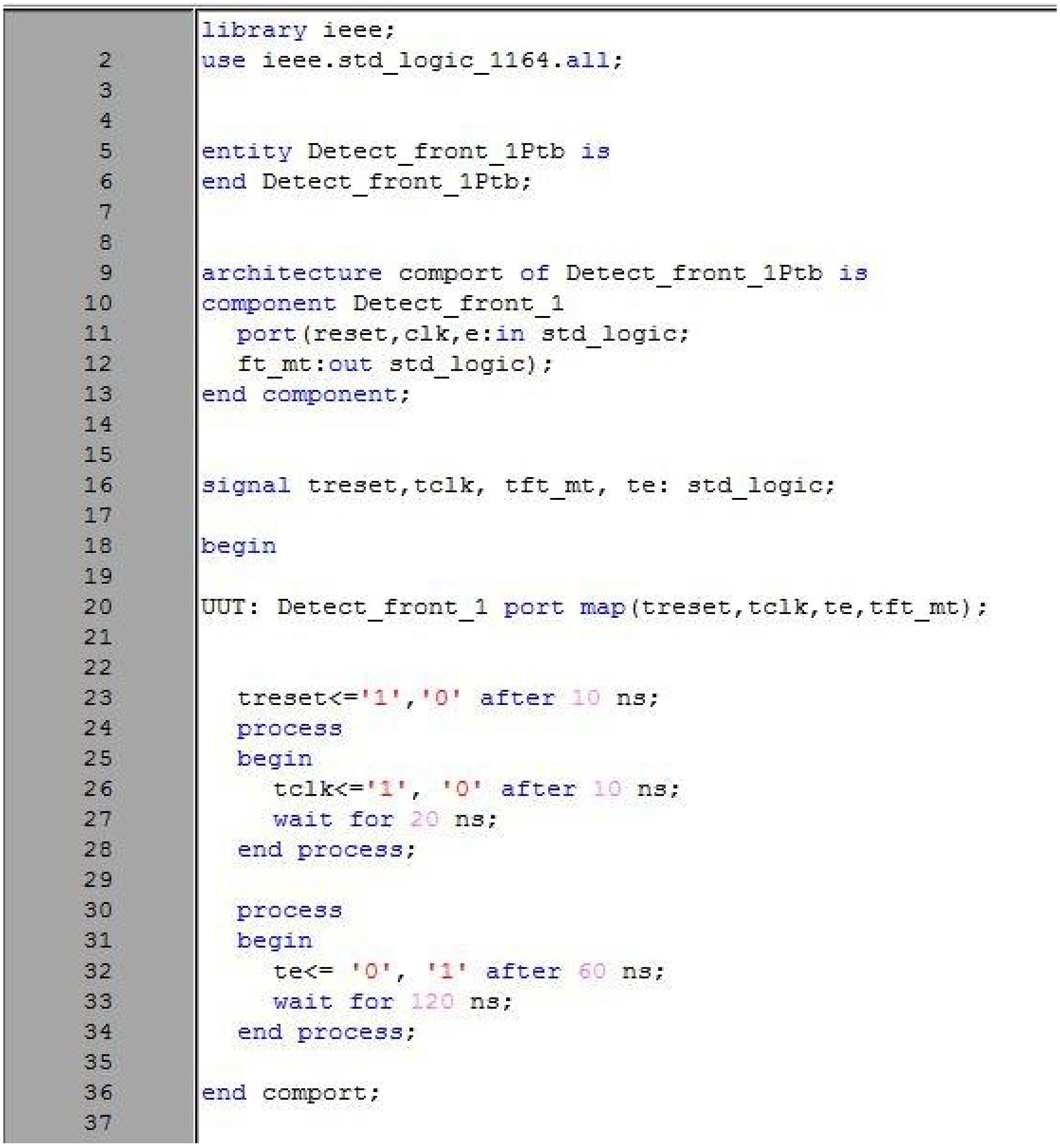
1. le composant Detect_front_1 et Detect_front_2

Programmation avec FSM:



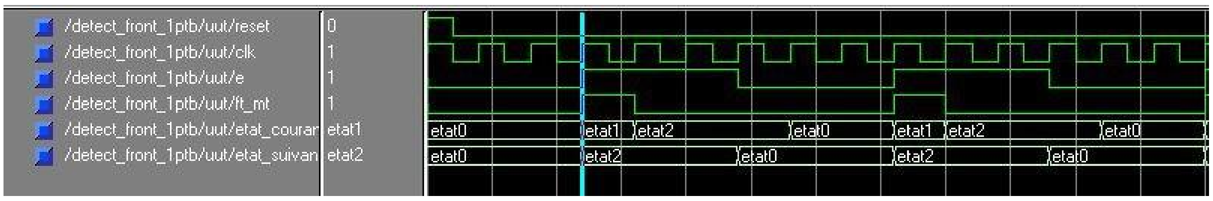
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity Detect_front_1 is
  port(reset,clk,e:in std_logic;
  ft_mt:out std_logic);
end Detect_front_1;

architecture comport of Detect_front_1 is
type state_type is (etat0,etat1,etat2);
signal etat_courant,etat_suivant:state_type;
begin
  process(etat_courant,e)
  begin
    case etat_courant is
      when etat0 => ft_mt <= '0';
            if(e='1') then etat_suivant<= etat1;
            else etat_suivant <= etat0;
            end if;
      when etat1 => ft_mt <= '1';
            etat_suivant<= etat2;
      when etat2 => ft_mt <= '0';
            if(e='0') then etat_suivant<= etat0;
            else etat_suivant <= etat2;
            end if;
    end case;
  end process;
  process(clk,reset)
  begin
    if (clk'event and clk='1') then
      if reset='1' then
            etat_courant<= etat0;
      else
            etat_courant <=etat_suivant;
      end if;
    end if;
  end process;
end comport;
```

Le test bench



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


entity Detect_front_1Ptb is
end Detect_front_1Ptb;


architecture comport of Detect_front_1Ptb is
component Detect_front_1
  port(reset,clk,e:in std_logic;
  ft_mt:out std_logic);
end component;


signal treset,tclk, tft_mt, te: std_logic;

begin

UUT: Detect_front_1 port map(treset,tclk,te,tft_mt);


  treset<='1','0' after 10 ns;
  process
  begin
     tclk<='1', '0' after 10 ns;
     wait for 20 ns;
  end process;

  process
  begin
     te<= '0', '1' after 60 ns;
     wait for 120 ns;
  end process;

end comport;

```

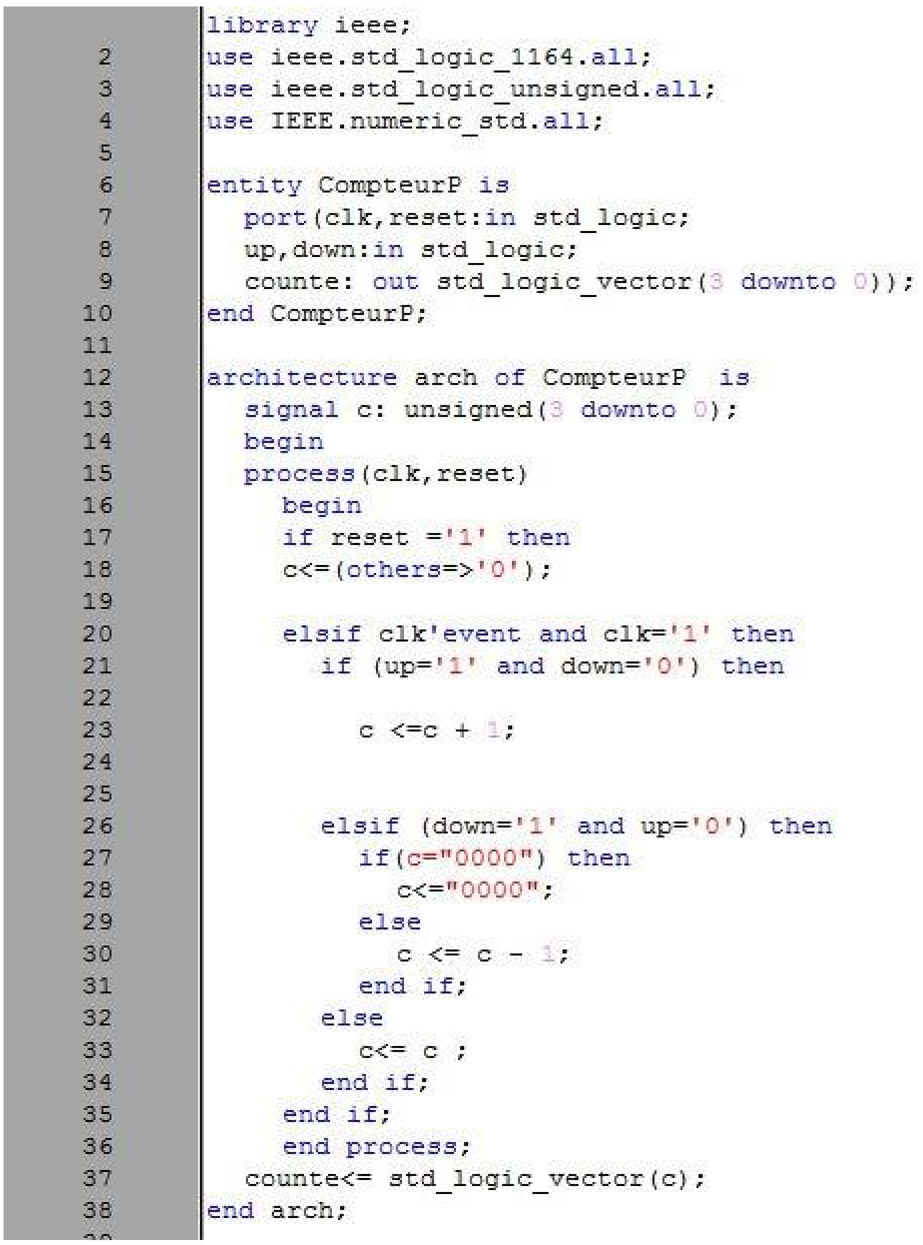
La Simulation



/detect_front_1ptb/uut/reset 0
/detect_front_1ptb/uut/clk 1
/detect_front_1ptb/uut/e 1
/detect_front_1ptb/uut/ft_mt 1
/detect_front_1ptb/uut/etat_courar etat1
/detect_front_1ptb/uut/etat_suivan etat2
etat0 etat1 etat2 etat0 etat1 etat2 etat0
etat0 etat2 etat0 etat2 etat0

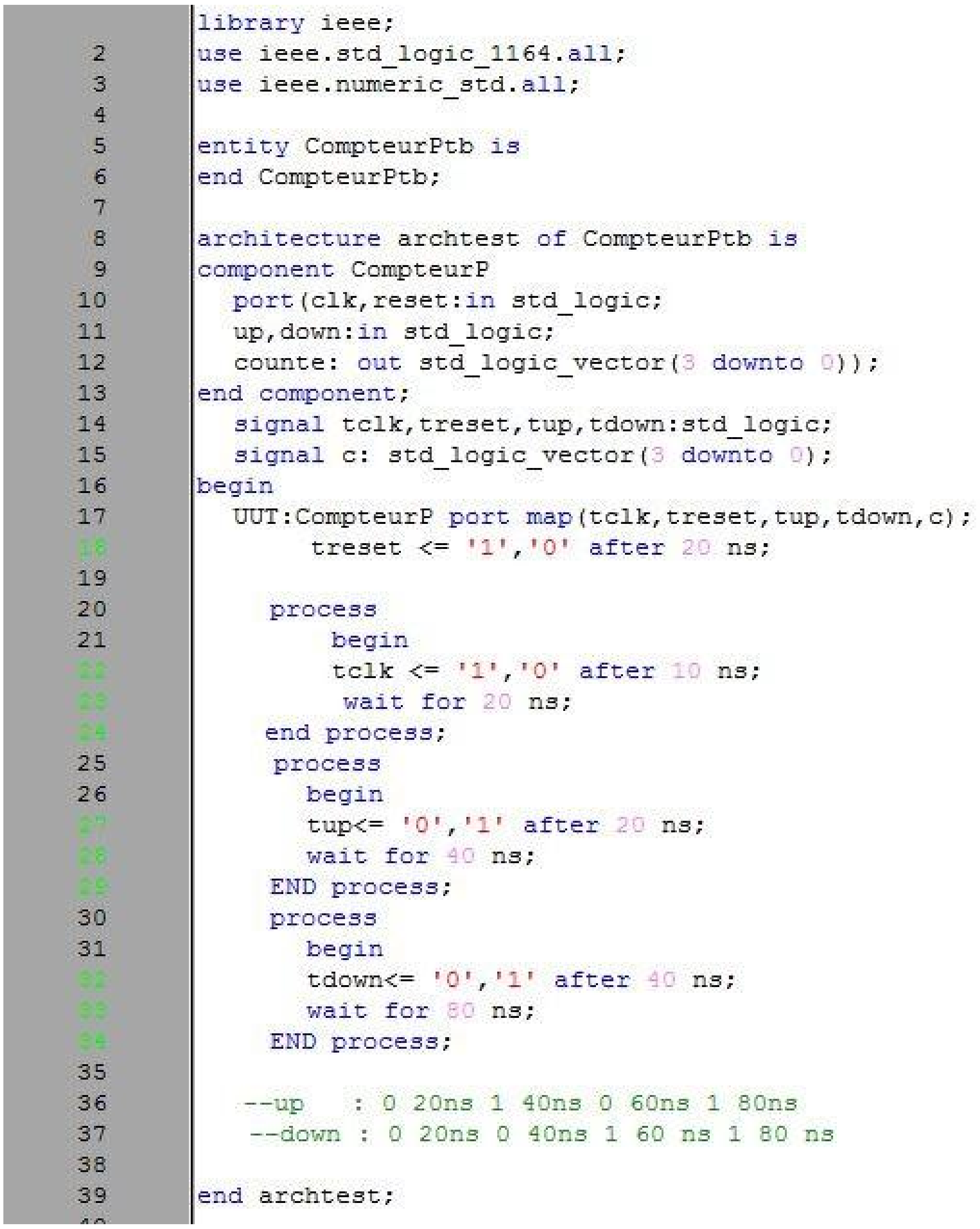
2. Le composant Compteur

La programmation



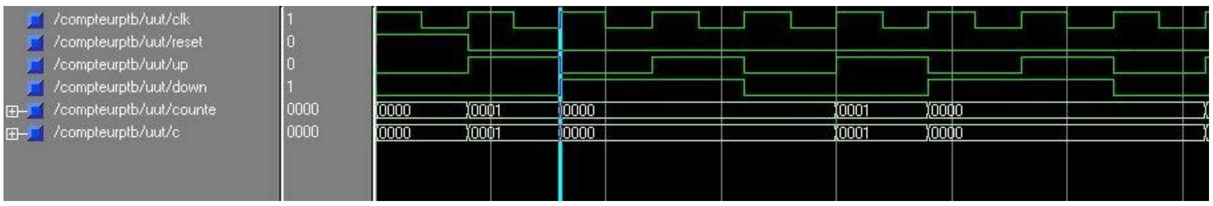
```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use IEEE.numeric_std.all;

entity CompteurP is
  port(clk,reset:in std_logic;
  up,down:in std_logic;
  counte: out std_logic_vector(3 downto 0));
end CompteurP;

architecture arch of CompteurP  is
  signal c: unsigned(3 downto 0);
  begin
  process(clk,reset)
    begin
    if reset ='1' then
    c<=(others=>'0');

    elsif clk'event and clk='1' then
      if (up='1' and down='0') then

        c <=c + 1;


      elsif (down='1' and up='0') then
        if(c="0000") then
          c<="0000";
        else
          c <= c - 1;
        end if;
      else
        c<= c ;
      end if;
    end if;
    end process;
  counte<= std_logic_vector(c);
end arch;
```

Le test bench



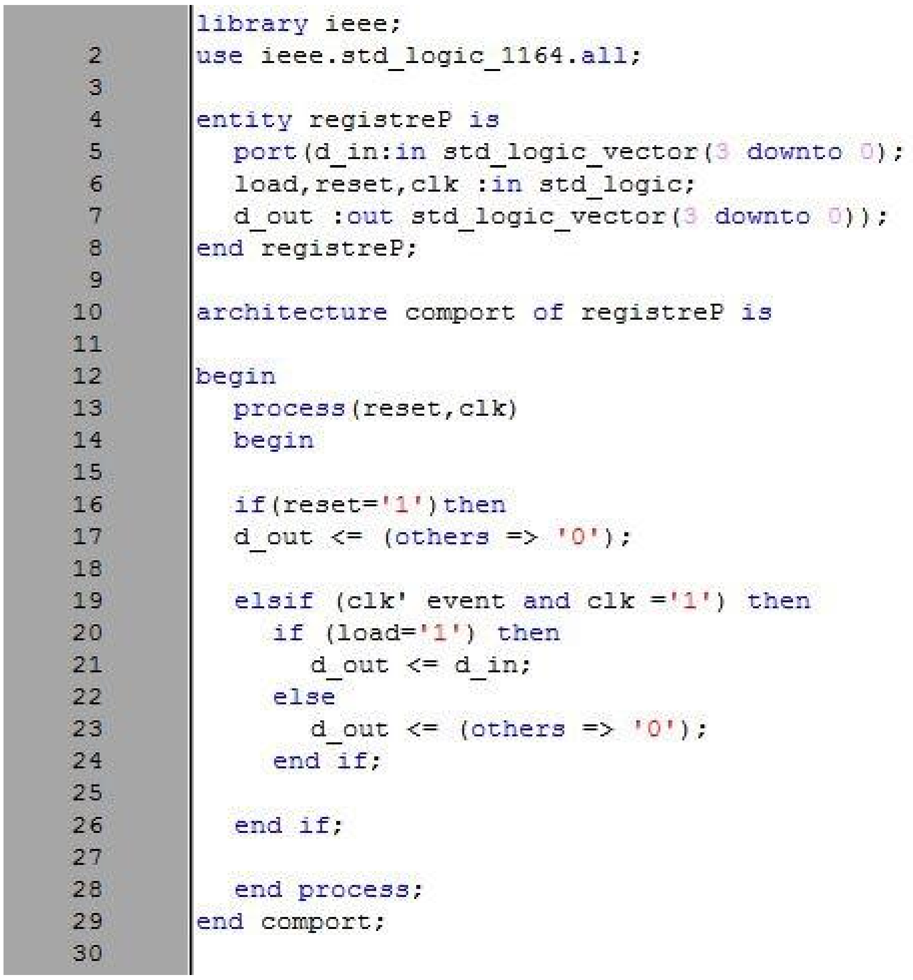
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity CompteurPtb is
end CompteurPtb;

architecture archtest of CompteurPtb is
component CompteurP
  port(clk,reset:in std_logic;
  up,down:in std_logic;
  counte: out std_logic_vector(3 downto 0));
end component;
  signal tclk,treset,tup,tdown:std_logic;
  signal c: std_logic_vector(3 downto 0);
begin
  UUT:CompteurP port map(tclk,treset,tup,tdown,c);
        treset <= '1','0' after 20 ns;

     process
         begin
         tclk <= '1','0' after 10 ns;
          wait for 20 ns;
     end process;
      process
        begin
        tup<= '0','1' after 20 ns;
        wait for 40 ns;
      END process;
      process
        begin
        tdown<= '0','1' after 40 ns;
        wait for 80 ns;
      END process;

    --up   : 0 20ns 1 40ns 0 60ns 1 80ns
     --down : 0 20ns 0 40ns 1 60 ns 1 80 ns

end archtest;
```

La Simulation



3. Le composant Registre

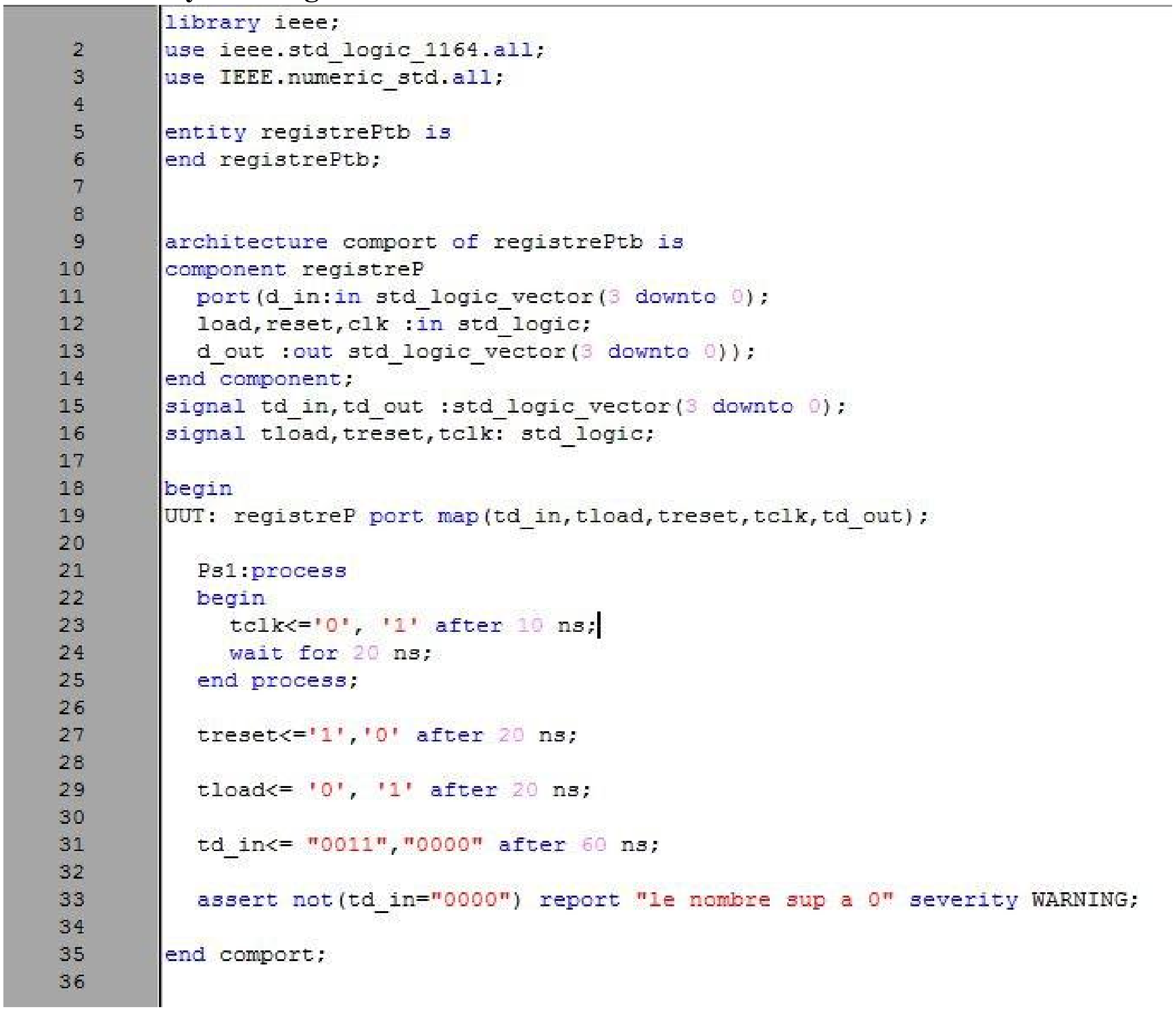
La programmation



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

entity registreP is
  port(d_in:in std_logic_vector(3 downto 0);
  load,reset,clk :in std_logic;
  d_out :out std_logic_vector(3 downto 0));
end registreP;

architecture comport of registreP is

begin
  process(reset,clk)
  begin

  if(reset='1')then
  d_out <= (others => '0');

  elsif (clk' event and clk ='1') then
    if (load='1') then
      d_out <= d_in;
    else
      d_out <= (others => '0');
    end if;

  end if;

  end process;
end comport;
```

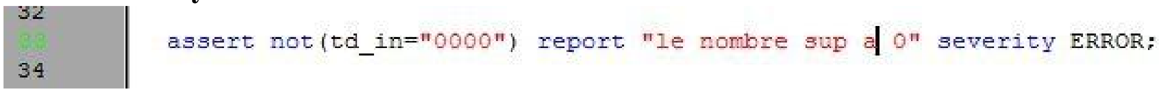
Le test bench

- **Severity warning**



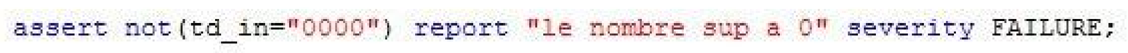
```
library ieee;
use ieee.std_logic_1164.all;
use IEEE.numeric_std.all;

entity registrePtb is
end registrePtb;


architecture comport of registrePtb is
component registreP
  port(d_in:in std_logic_vector(3 downto 0);
  load,reset,clk :in std_logic;
  d_out :out std_logic_vector(3 downto 0));
end component;
signal td_in,td_out :std_logic_vector(3 downto 0);
signal tload,treset,tclk: std_logic;

begin
UUT: registreP port map(td_in,tload,treset,tclk,td_out);

  Ps1:process
  begin
    tclk<='0', '1' after 10 ns;
    wait for 20 ns;
  end process;

  treset<='1','0' after 20 ns;

  tload<= '0', '1' after 20 ns;

  td_in<= "0011","0000" after 60 ns;

  assert not(td_in="0000") report "le nombre sup a 0" severity WARNING;

end comport;
```

- **Severity error :**



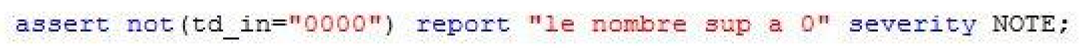
```
  assert not(td_in="0000") report "le nombre sup a 0" severity ERROR;
```

- **Severity failure**



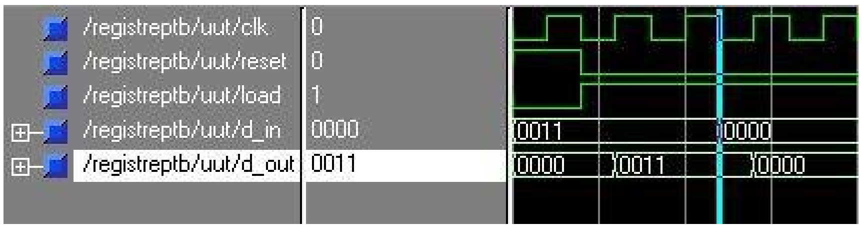
```
assert not(td_in="0000") report "le nombre sup a 0" severity FAILURE;
```

- **Severity NOTE**

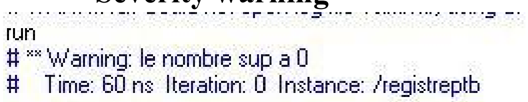


```
assert not(td_in="0000") report "le nombre sup a 0" severity NOTE;
```

La Simulation

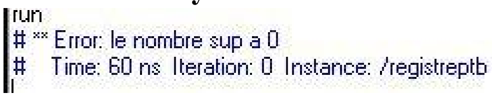


Severity warning



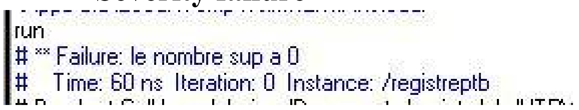
```
run
# ** Warning: le nombre sup a 0
#    Time: 60 ns  Iteration: 0  Instance: /registreptb
```

Severity error



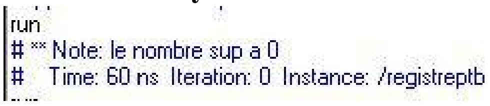
```
run
# ** Error: le nombre sup a 0
#    Time: 60 ns  Iteration: 0  Instance: /registreptb
```

Severity failure



```
run
# ** Failure: le nombre sup a 0
#    Time: 60 ns  Iteration: 0  Instance: /registreptb
```

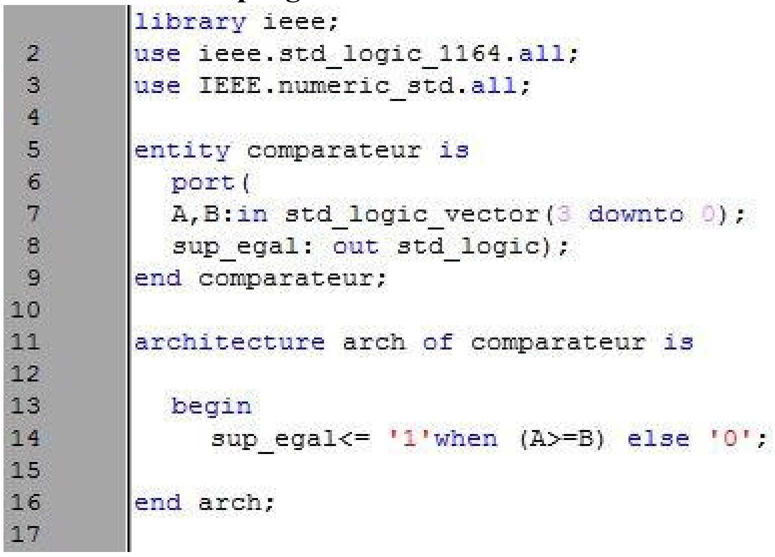
Severity Note



```
run
# ** Note: le nombre sup a 0
#    Time: 60 ns  Iteration: 0  Instance: /registreptb
```

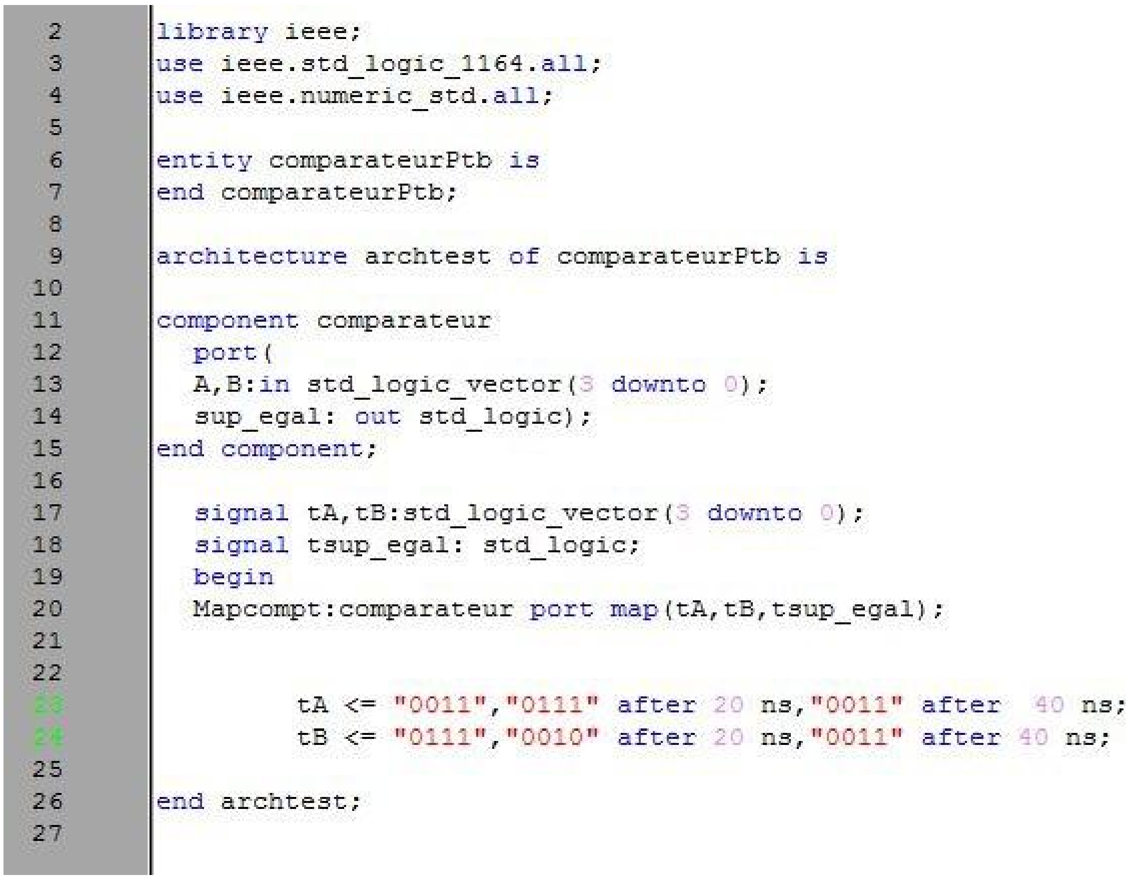
4. Le composant comparateur

La programmation



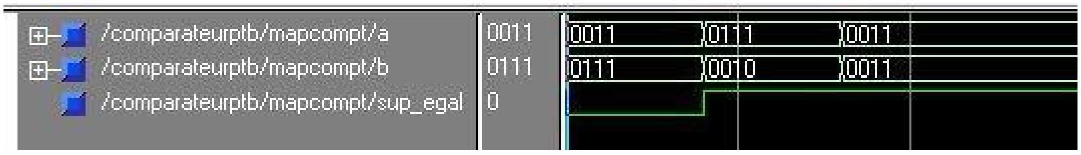
```vhdl
library ieee;
use ieee.std_logic_1164.all;
use IEEE.numeric_std.all;

entity comparateur is
   port(
   A,B:in std_logic_vector(3 downto 0);
   sup_egal: out std_logic);
end comparateur;

architecture arch of comparateur is

   begin
      sup_egal<= '1'when (A>=B) else '0';

end arch;
```

Le test bench



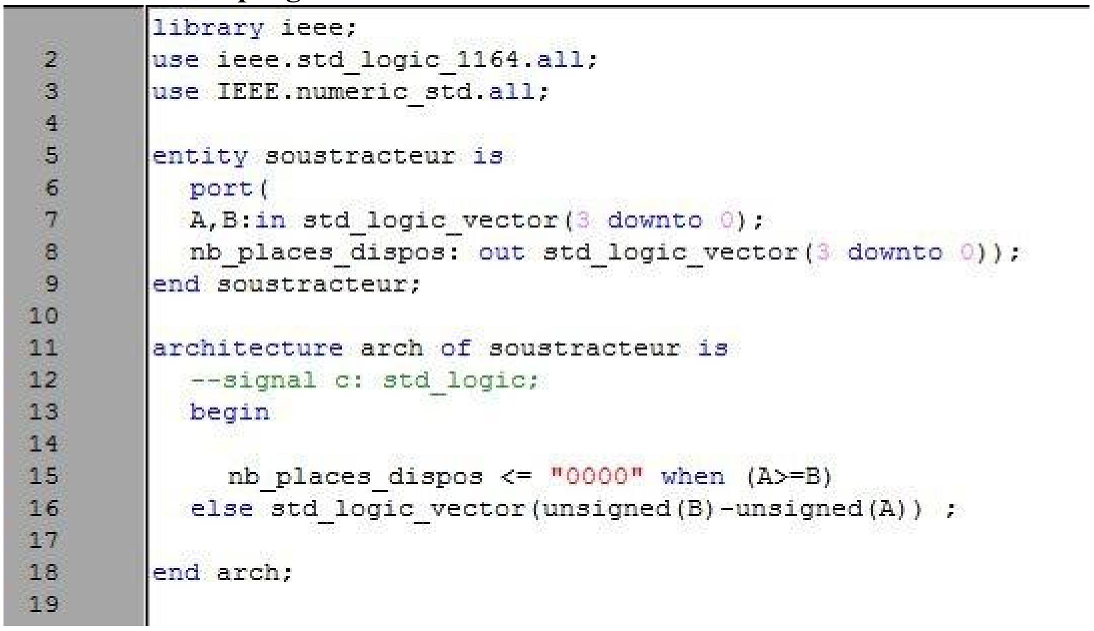
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity comparateurPtb is
end comparateurPtb;

architecture archtest of comparateurPtb is

component comparateur
  port(
  A,B:in std_logic_vector(3 downto 0);
  sup_egal: out std_logic);
end component;

  signal tA,tB:std_logic_vector(3 downto 0);
  signal tsup_egal: std_logic;
  begin
  Mapcompt:comparateur port map(tA,tB,tsup_egal);


         tA <= "0011","0111" after 20 ns,"0011" after  40 ns;
         tB <= "0111","0010" after 20 ns,"0011" after 40 ns;

end archtest;
```

La simulation



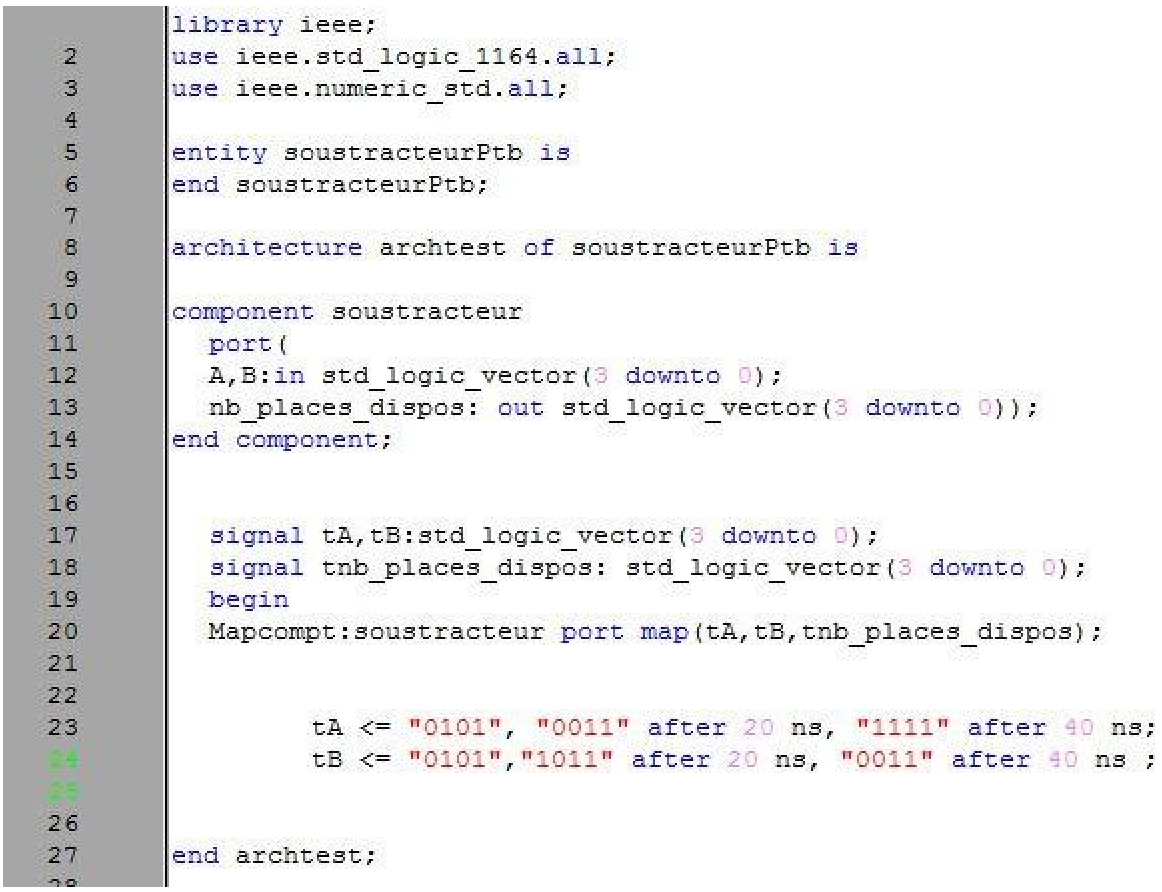
5. Le composant soustracteur

La programmation



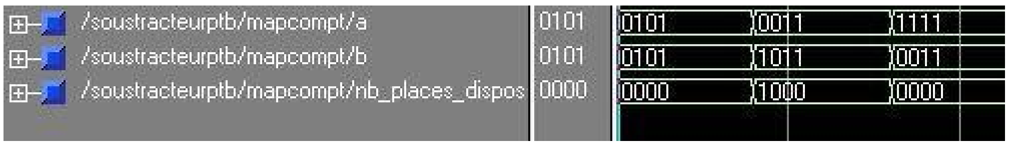
```
library ieee;
use ieee.std_logic_1164.all;
use IEEE.numeric_std.all;

entity soustracteur is
  port(
  A,B:in std_logic_vector(3 downto 0);
  nb_places_dispos: out std_logic_vector(3 downto 0));
end soustracteur;

architecture arch of soustracteur is
  --signal c: std_logic;
  begin

    nb_places_dispos <= "0000" when (A>=B)
  else std_logic_vector(unsigned(B)-unsigned(A)) ;

end arch;
```

Le test bench



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

entity soustracteurPtb is
end soustracteurPtb;

architecture archtest of soustracteurPtb is

component soustracteur
  port(
  A,B:in std_logic_vector(3 downto 0);
  nb_places_dispos: out std_logic_vector(3 downto 0));
end component;


  signal tA,tB:std_logic_vector(3 downto 0);
  signal tnb_places_dispos: std_logic_vector(3 downto 0);
  begin
  Mapcompt:soustracteur port map(tA,tB,tnb_places_dispos);


        tA <= "0101", "0011" after 20 ns, "1111" after 40 ns;
        tB <= "0101","1011" after 20 ns, "0011" after 40 ns ;


end archtest;
```

La simulation


/soustracteurptb/mapcompt/a
0101
0101
0011
1111
/soustracteurptb/mapcompt/b
0101
0101
1011
0011
/soustracteurptb/mapcompt/nb_places_dispos
0000
0000
1000
0000