

Exercice 1 avec « & »

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity additionneur is
  Port (
    A : in STD_LOGIC_VECTOR(5 downto 0);
    B : in STD_LOGIC_VECTOR(3 downto 0);
    S : out STD_LOGIC_VECTOR(6 downto 0)
  );
end additionneur;

architecture comportement of additionneur is
  signal A_signed : signed(5 downto 0);
  signal B_signed_etendu : signed(5 downto 0);
  signal Somme_signed : signed(6 downto 0);
begin

  A_signed <= signed(A(5) & A);
  B_signed_etendu <= signed(B(3) & B(3) & B(3) & B);

  Somme_signed <= A_signed + B_signed;

  S <= std_logic_vector(Somme_signed);

end comportement;
```

Exercice 1 avec « resize »

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity additionneur is
  Port (
    A : in STD_LOGIC_VECTOR(5 downto 0);
    B : in STD_LOGIC_VECTOR(3 downto 0);
    S : out STD_LOGIC_VECTOR(6 downto 0)
  );
end additionneur;

architecture comportement of additionneur is
  signal A_signed : signed(5 downto 0);
  signal B_signed_etendu : signed(5 downto 0);
  signal Somme_signed : signed(6 downto 0);
begin

  A_signed <= signed(A);
  B_signed_etendu <= resize(signed(B), 6);

  Somme_signed <= resize(A_signed, 7) + resize(B_signed_etendu, 7);

  S <= std_logic_vector(Somme_signed);

end comportement;
```

Remarque : on pouvait aussi passer par les *integer* (sans faire de *resize* ni *&*)

Exercice 2

Compteur/comparateur

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity jeu is
  Port (
    clk : in STD_LOGIC;
    E : in STD_LOGIC_VECTOR(2 downto 0);
    P : in STD_LOGIC;
    G : out STD_LOGIC
  );
end jeu;

architecture comportement of jeu is
  signal Compteur : integer range 0 to 7 := 0;
  signal clk_div : STD_LOGIC := '0';
  signal div_Compteur : integer := 0;

begin
  process (clk)
  begin
    if (clk'event and clk='1') then
      if div_Compteur = 12499999 then
        div_Compteur <= 0;
        clk_div <= not clk_div;
      else
        div_Compteur <= div_Compteur + 1;
      end if;
    end if;
  end process;

  process (clk_div)
  begin
    if (clk'event and clk='1') then
      if P = '0' then
        if Compteur = 7 then
          Compteur <= 0;
        else
          Compteur <= Compteur + 1;
        end if;
      end if;
    end if;
  end process;

  process (Compteur, E)
  begin
    if Compteur = to_integer(unsigned(E)) then
      G <= '1';
    else
      G <= '0';
    end if;
  end process;

end comportement;
```

Exercise 3

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity disco is
  Port (
    clk : in STD_LOGIC;
    B : in STD_LOGIC;
    s : out STD_LOGIC_VECTOR(2 downto 0);
  );
end disco;

architecture comportement of disco is
  type state_type is (E0, E1, E2, E3, E4, E5, E6, E7, E8);
  signal state, next_state : state_type;

begin
  process (state, B)
  begin
    case state is
      when E0 =>
        if B = '0' then
          next_state <= E1;
        else
          next_state <= E0;
        end if;
      when E1 =>
        if B = '0' then
          next_state <= E2;
        else
          next_state <= E1;
        end if;
      when E2 =>
        if B = '0' then
          next_state <= E3;
        else
          next_state <= E2;
        end if;
      when E3 =>
        if B = '0' then
          next_state <= E4;
        else
          next_state <= E3;
        end if;
      when E4 =>
        if B = '0' then
          next_state <= E5;
        else
          next_state <= E4;
        end if;
      when E5 =>
        if B = '0' then
          next_state <= E6;
        else
          next_state <= E5;
        end if;
      when E6 =>
        if B = '0' then
          next_state <= E7;
        else
          next_state <= E6;
        end if;
      when E7 =>
        if B = '0' then
          next_state <= E8;
        else
          next_state <= E7;
        end if;
      when E8 =>
        if B = '0' then
          next_state <= E0;
        else
          next_state <= E8;
        end if;
    end case;
    state <= next_state;
  end process;
end;
```

```

        next_state <= E6;
    end if;
when E7 =>
    if B = '0' then
        next_state <= E8;
    else
        next_state <= E7;
    end if;
when E8 =>
    if B = '0' then
        next_state <= E0;
    else
        next_state <= E8;
    end if;
when others =>
    next_state <= E0;
end case;
end process;

process (clk)
begin
    if (clk'event and clk='1') then
        state <= next_state;
    end if;
end process;

process (state)
begin
    case state is
        when E0 =>
            s <= "000";
        when E1 =>
            s <= "100";
        when E2 =>
            s <= "010";
        when E3 =>
            s <= "001";
        when E4 =>
            s <= "111";
        when E5 =>
            s <= "000";
        when E6 =>
            s <= "111";
        when E7 =>
            s <= "000";
        when E8 =>
            s <= "111";
        when others =>
            s <= "000";
    end case;
end process;
end comportement;
```