Exercices

Utilisation du package numeric_std

1. Réaliser un additionneur 4 bits en instanciant l'additionneur 1 bit déjà présenté
2. Décrire un additionneur 4 bits en utilisant le package `numeric_std`
3. Décrire un soustracteur 4 bits en utilisant le package `numeric_std`
4. Décrire un comparateur 8 bits en utilisant le package `numeric_std`
5. Pour le comparateur 8 bits, est-il nécessaire de faire des conversions en `unsigned` ?

Exercice 1

Solution

```
library ieee;
use ieee.std_logic_1164.all;
entity additionneur4bit is
port(
     a,b       : in  std_logic_vector(3 downto 0);
     cin       : in  std_logic;
     s         : out std_logic_vector(3 downto 0);
     cout      : out std_logic);
end;

architecture archConc of additionneur4bit is
component additionneur
port(
     a,b,cin   : in  std_logic;
     s,cout    : out std_logic);
end component;
signal c : std_logic_vector(2 downto 0);
begin
   add_1: additionneur port map (a(0),b(0),cin,s(0),c(0));
```

Exercice 1

Solution

```
   add_2: additionneur port map (a(1),b(1),c(0),s(1),c(1));
   add_3: additionneur port map (a(2),b(2),c(1),s(2),c(2));
   add_4: additionneur port map (a(3),b(3),c(2),s(3),cout);
end archConc;
```

Exercice 2

Solution

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity additionneur4bit_1 is
port(
     a,b       : in  std_logic_vector(3 downto 0);
     cin       : in  std_logic;
     s         : out std_logic_vector(3 downto 0);
     cout      : out std_logic);
end;

architecture archConc of additionneur4bit_1 is
   signal s_int : unsigned (4 downto 0);
   signal cin_nat: natural range 0 to 1;
begin
   cin_nat <= 1 when cin='1' else 0;
   s_int <= unsigned('0' & a) + unsigned('0' & b) + cin_nat;
   s     <= std_logic_vector(s_int(3 downto 0));
   cout  <= s_int(4);
```

Exercice 2

Solution

```
end archConc;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity additionneur4bit_2 is
port(
     a,b       : in  std_logic_vector(3 downto 0);
     cin       : in  std_logic;
     s         : out std_logic_vector(3 downto 0);
     cout      : out std_logic);
end;

architecture archConc of additionneur4bit_2 is
   signal s_int : unsigned (4 downto 0);
   signal cin_vector: std_logic_vector (0 downto 0);
begin
   cin_vector(0) <= cin;
```

Exercice 2

Solution

```
    s_int <= unsigned('0' & a) + unsigned('0' & b) + unsigned("0000" &
      cin_vector);
    --s_int <= resize(unsigned(a),5) + resize(unsigned(b),5) +
      resize(unsigned(cin_vector),5);
    s     <= std_logic_vector(s_int(3 downto 0));
    cout  <= s_int(4);
end archConc;
```

Exercice 3

Solution

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity soustracteur4bit is
port(
     a,b       : in  std_logic_vector(3 downto 0);
     cin       : in  std_logic;
     s         : out std_logic_vector(3 downto 0);
     cout      : out std_logic);
end;

architecture archConc of soustracteur4bit is
   signal s_int : unsigned (4 downto 0);
   signal cin_vector: std_logic_vector (0 downto 0);
begin
   cin_vector(0) <= cin;
   s_int <= unsigned('0' & a) + unsigned(not('0' & b)) +
     unsigned("0000" & cin_vector);
   --s_int <= resize(unsigned(a),5) + resize(unsigned(b),5) +
     resize(unsigned(cin_vector),5);
```

Exercice 3

Solution

```
    s      <= std_logic_vector(s_int(3 downto 0));
    cout   <= s_int(4);
end archConc;
```

Exercice 4

Solution

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity comparateur8bit_1 is
port(
     a,b       : in  std_logic_vector(7 downto 0);
     eq,gt,lt  : out  std_logic);
end;

architecture archConc of comparateur8bit_1 is
   signal diff: unsigned (7 downto 0);
begin
   diff <= unsigned(a)-unsigned(b);
   eq <= '1' when diff = 0 else '0';
   gt <= '1' when diff > 0 else '0';
   lt <= '1' when diff < 0 else '0';
end archConc;
```

Exercice 4

Solution

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity comparateur8bit is
port(
     a,b       : in  std_logic_vector(7 downto 0);
     eq,gt,lt  : out  std_logic);
end;

architecture archConc of comparateur8bit is
begin
   eq <= '1' when a = b else '0';
   gt <= '1' when a > b else '0';
   lt <= '1' when a < b else '0';
end archConc;
```

Exercices de synthèse

Codage d'une ALU simple

1. Donnez un code permettant de synthétiser une ALU opérant sur des données de 8 bits et possédant les 5 modes suivants : `ADD; SUB; AND; OR; XOR`
2. Modifiez le code pour que la taille des données soit générique
3. Observez le résultat de synthèse dans la vue RTL
4. Écrivez et simulez cette ALU avec un testbench permettant de vérifier les 5 modes sur 4 vecteurs d'entrée chacun