Exemple de code VHDL pour les nombres complexes

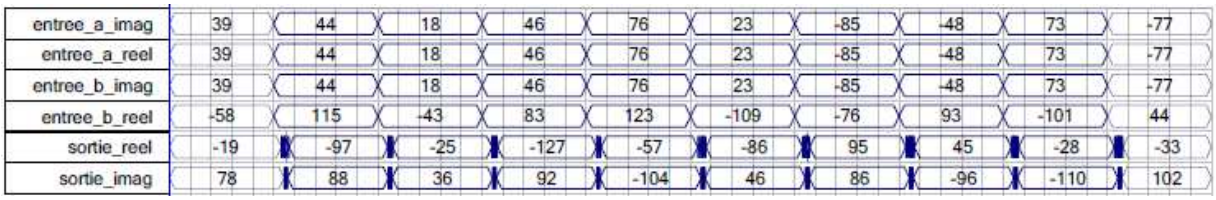
Dans ce document, nous allons parler de la manipulation des nombres complexes. Il sera question d'implémenter une addition et une multiplication pour les nombres complexes. Un nombre complexe sera normalement spécifié dans un système comme étant deux vecteurs : un vecteur pour la partie réelle et un vecteur pour la partie imaginaire.

Dans le cas d'une addition, il y aura deux nombres en entrée ce qui représente quatre vecteurs. En sortie, il y aura un vecteur réel et un vecteur imaginaire. L'entité sera donc déclarée comme ceci :

```
ENTITY complex IS
    PORT (
          entree_a_reel : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          entree_a_imag : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          entree_b_reel : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          entree_b_imag : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          sortie_reel   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
          sortie_imag   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
         );
END complex;
```

L'addition de nombre complexes est simple : il faut additionner les nombres réels ensemble et les nombres imaginaires ensembles. La somme des réels représentera la partie réelle de la sortie tandis que la somme des nombres imaginaires sera la partie imaginaire de la sortie. Le code en entier ressemble à ceci :

```
LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY complex IS
    PORT (
          entree_a_reel : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          entree_a_imag : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          entree_b_reel : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          entree_b_imag : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          sortie_reel   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
          sortie_imag   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
         );
END complex;

ARCHITECTURE rtl OF complex IS
BEGIN

sortie_reel <= entree_a_reel + entree_b_reel;
sortie_imag <= entree_a_imag + entree_b_imag;

END;
```



Pour faire une multiplication, le niveau de difficulté augmente un peu. Nous savons que la multiplication de *(a+jb)* et *(c+jd)* donnerait *(ac-bd)+j(ad+bc)*. Il suffirait donc de mettre la partie réelle de la sortie égale à *(ac-bd)* et la partie imaginaire égale à *(ad+bc)*.

Les problèmes surviennent quand on pense aux signes. L'addition est une opération qui reste pareille peu importe que les nombres soient en complément a 2 ou s'ils ne sont pas signes. La multiplication, cependant, ne bénéficie pas de la même caractéristique. Avec nos déclarations normales, nous utilisons IEEE.STD_LOGIC_UNSIGNED.ALL. Pour tenir compte des signes il faut utiliser IEEE.STD_LOGIC_SIGNED.ALL à la place.

```
LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_SIGNED.ALL;

ENTITY complex IS
    PORT (
          entree_a_reel : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          entree_a_imag : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          entree_b_reel : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          entree_b_imag : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          sortie_reel   : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
          sortie_imag   : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
         );
END complex;

ARCHITECTURE rtl OF complex IS

BEGIN

sortie_reel <= (entree_a_reel * entree_b_reel)-(entree_a_imag *
entree_b_imag);
sortie_imag <= (entree_a_reel* entree_b_imag) + (entree_a_imag *
entree_b_reel);

END;
```