# TP n° 5 : Utilisation du processeur NIOS II avec instructions «spécialisées»

## 0. Introduction

L'objectif de ce TP est d'utiliser le processeur NIOS II (cœur logiciel) sur une carte FPGA d'Altera pour exécuter des programmes C et mesurer leurs temps d'exécution.
On examinera comment

- compiler une version du processeur à l'aide du logiciel Quartus II
- charger le processeur et l'environnement associé dans le FPGA (charger le fichier de configuration du processeur sur la carte à l'aide de la connexion USB – Blaster)
- charger un programme C dans la mémoire du processeur (charger le fichier correspondant au programme dans le FPGA) et l'exécuter
- écrire des programmes C sous l'environnement IDE, les charger dans le FPGA et les exécuter
- ajouter des instructions spécialisées au processeur NIOS
- mesurer les temps d'exécution des programmes.

## 1. Développement d'un programme sous l'environnement IDE

Ouvrir le logiciel NIOS IDE
Ouvrir une nouvelle application C : *new* | *CC++ Application* et choisir *Hello_world*. Choisir pour SOC Builder System *C : \niosII_cyclone_1c20\quartus_project\system.ptf*
Activer le projet en cliquant dessus.
Dans le menu *Projet* | *Properties* | *C/C++ Build* , sélectionner la configuration *Release* et un niveau d'optimisation (-O2 ou –O3)
Puis faire *Build Projet*.
Puis exécuter le projet avec *Run* | *Run as* | *Nios Hardware*

## 2. Test des instructions spécialisées ajoutées au processeur NIOS

Dans le TP 4, on a réalisé des instructions spécialisées :

- VABS (a, b) : calcul de la valeur absolue de la différence a-b
- VABS4 (a,b) : calcul SIMD de la valeur absolue de la différence de 4x4 octets non signés
- SVABS (a,b) : calcul de la somme des valeurs absolues des différences de 4x4 octets non signés
- S2PS (a, b) : somme des produits SIMD de 2x16 bits non signés.
- ADDS (a,b) : addition saturée de deux entiers signés
- ADDUS(a,b) : addition saturée de deux entiers non signés
- ADDHS (a,b) : addition saturée SIMD de 2x16 bits signés
- ADDHUS(a,b) : addition saturée SIMD de 2x16 bits non signés
- ADDBUS(a,b) : addition saturée SIMD de 4x8 bits non signés

Ecrire et exécuter un programme pour tester le fonctionnement correct de ces instructions.

## 3. Algorithme d'estimation de mouvement

Soit l'algorithme suivant pour l'estimation de mouvement

```
for(l=0; l<nVERT; l++)
     for(k=0, c=0; c<nHORZ; k=+8, c++){
             answer = 0;
             for(j=0; j<16; j++)
                  for(i=0; i<16; i++)
                      answer += abs(x[l+j][k+i] – y[l+j][k+i]);
            result[l][c] = answer;}
```

**Programme C initial**

Insérer dans le programme les mesures de temps. Exécuter le programme pour nVERT=nHORZ=256.

Mesurer le temps d'exécution (en cycles par point) pour la version initiale du programme en utilisant deux versions :

- fonction abs qui renvoie a-b si a>b et b-a sinon
- #define abs (a-b) ((a>b) ? (a-b) : (b-a))
- Instruction spécialisée VABS (a,b) définie dans le TP4

**Programme C avec instruction spécialisée SVABS4**

Modifier le programme en utilisant l'instruction SVABS4.

Mesurer le temps d'exécution et calculer l'accélération.

## 4. Produit scalaire

Soit le programme suivant

```
unsigned short X[N], Y[N] ;
int i, S=0 ;
for (i=0 ; i<N ; i++)
      S+=X[i]*Y[i];
```

**Programme C initial**

Mesurer le temps d'exécution du programme C.

**Programme C avec instruction spécialisée**

Modifier le programme en utilisant l'instruction S2PS.

Mesurer le temps d'exécution et calculer l'accélération.

**NB : pour tous les programmes, on vérifiera que les résultats d'exécution sont corrects.**

## 5. Annexes (à rajouter)

**Mesure des temps d'exécution**

**Chargement d'une configuration sur le FPGA (processeurs + opérateurs spécialisés)**

**Modification du processeur et ajout de nouvelles instructions.**