TP n° 4 : Réalisation d'instructions SIMD pour le processeur NIOS II

0. Introduction

On peut définir de nouvelles instructions (« customization ») pour le processeur NIOS II (cœur logiciel pour FPGA d'Altera), selon le schéma

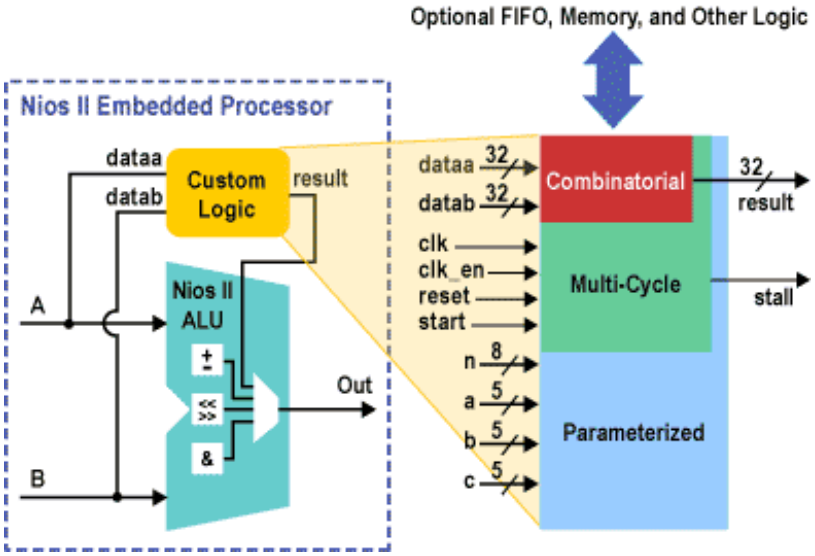

Figure 1 : Spécialisation d'instructions pour le processeurs NIOS II

L'objectif de ce TP est de définir des instructions SIMD pour

- le traitement d'images pour accélérer l'exécution de la fonction estimation de mouvement
- le traitement du signal pour accélérer l'exécution du produit scalaire sur des vecteurs de type « short »
- l'arithmétique saturée.

Dans ce TP, on examinera comment

- Ecrire les programmes VHDL permettant de réaliser les circuits implantant les différentes instructions
- Compiler le code VHDL avec le programme Quartus II
- Tester les circuits obtenus à l'aide de la simulation logique.

1. Opérateurs Valeur absolue

Opérateur vabs (a-b)

L'opérateur vabs est implanté en C à l'aide d'une fonction. Dans une première étape, on veut réaliser un opérateur valeur absolue sur la différence de deux entiers non signés 32 bits. Définir le code VHDL permettant d'implanter le matériel pour une instruction spécialisée de ce type. Vérifier le fonctionnement correct par simulation.

Opérateur simd vabs4 sur des octets non signés

On veut réaliser l'opérateur simd vabs4(a3-b3|a2-b2|a1-b1|a0-b0) qui calcule la valeur absolue de la différence d'octets non signés à partir de deux mots de 32 bits contenant respectivement a3a2a1a0 et b3b2b1b0 et place les résultats dans un mot de 32 bits contenant v3|v2|v1|v0
avec v3 = |a3-b3|, v2 = |a2-b2|, v1= |a1-b1| et v0=|a0-b0|
Définir le code VHDL pour l'implantation de ces différentes instructions et vérifier le fonctionnement correct par simulation logique.

Opérateur somme des valeurs absolues des différences d'octets non signés

On veut réaliser l'opérateur qui calcule S = |a3-b3| + |a2-b2| + |a1-b1| + |a0-b0| à partir de deux mots de 32 bits contenant respectivement a3a2a1a0 et b3b2b1b0 et place les résultats dans un mot de 32 bits.
Définir le code VHDL pour l'implantation de ces différentes instructions et vérifier le fonctionnement correct par simulation logique.

2. Calcul SIMD du produit scalaire

On veut réaliser l'opération S = X1Y1 + X0Y0, dans lesquels Xi et Yi sont les éléments de vecteurs d'entiers non signés de 16 bits (unsigned short), tels que X1X0 (respectivement Y1Y0) sont les parties haute et basse d'un mot de 32bits.
Définir le code VHDL pour l'implantation de ces différentes instructions et vérifier le fonctionnement correct par simulation logique.

3. Opérations arithmétiques saturées

Définir le code VHDL pour l'implantation de l'arithmétique saturée sur NIOS II

- Addition scalaire 32 bits sur entiers signés
- Addition scalaire 32 bits sur entiers non signés
- Addition SIMD 2x16 bits sur entiers signés
- Addition SIMD 2x16 bits sur entiers non signés
- Addition SIMD 4x8 bits sur entiers non signés

Optionnel : définir les opérations de soustraction.

4. Annexe : utilisation de Quartus II

Compilation de code VHDL

Pour compiler un code VHDL, les étapes sont les suivantes :

- Ouvrir un nouveau projet à l'aide de *New Project Wizard*, en définissant un répertoire pour le projet, un nom de projet et le nom de l'entité la plus élevée dans la hiérarchie (ces deux derniers noms doivent être identiques).
- S'assurer que le projet contient bien le fichier .vhd à compiler
- Lancer la compilation dans le menu *Processing* | *start compilation*

Simulation

Lorsque le code VHDL est compilé, on peut le simuler l'exécution du circuit correspondant.
Les étapes sont les suivantes :
Génération des configurations d'entrée pour la simulation

- Ouvrir *New* | *Other files* | *Vector Waveform File*
- Aller dans le menu *Edit* |*Insert Node or Bus* | *Node Finder* | *List* : Vous obtenez alors à gauche la liste des signaux correspondant aux entrées et sortie du circuit en cours de conception
- Sélectionner dataa, datab et result à envoyer dans la fenêtre de droite à l'aide de la flèche >. Appuyer sur OK, puis OK après apparition d'une nouvelle fenêtre. Vous obtenez alors les formes de signaux correspondants. Sauvegarder le fichier .vwf sous le même nom que le nom de projet.
- Avec un clic à droite (de la souris) sur les signaux d'entrée, dataa ou datab, vous pouvez leur affecter des valeurs. Cliquez sur *Value* pour affecter des valeurs. Deux méthodes sont utiles : *arbitrary value* (fournir 8 chiffres hexadécimaux) ou *count value* (valeur de départ, incrément sur la première fenêtre, puis « count every » pour déterminer la fréquence du comptage (utilisez 100 ns pour pouvoir voir les 8 chiffres hexadécimaux).
- Attention, il faut souvent, après le clic droit sur les signaux, cliquer sur *zoom* | *fit in window* pour obtenir l'évolution complète des signaux du début à la fin de la simulation.

Simulation logique

- On peut alors simuler le circuit à l'aide du menu *processing* | *start simulation.*
- Les résultats sont obtenus dans le rapport de simulation.

5. Documents fournis

Les fichiers VHDL de certains des opérateurs sont fournis (site Web de l'option).