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

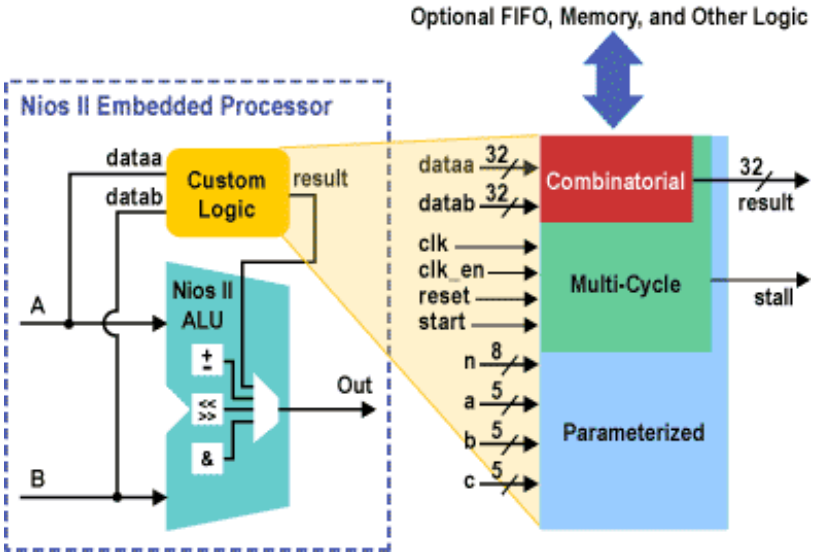

Figure 1 : Spécialisation d'instructions pour le processeurs NIOS II

L'objectif de ce TP est de définir des instructions SIMD pour le traitement d'images pour accélérer l'exécution des filtres 3x3 min, Gauss et médian.
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 MIN

Opérateur min

L'opérateur min est implanté en C à l'aide d'une fonction. Dans une première étape, on veut réaliser un opérateur MIN sur des 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.

Filtre MIN 3 x 3 pour le traitement d'images.

On veut réaliser les opérateurs matériels pour des instructions SIMD permettant de traiter simultanément 4 octets non signés (pixels) pour le filtre MIN. Les instructions à définir sont

- Décalage gauche d'un octet pour obtenir le mot de 32 bits pour les pixels X[i][j-1].
- Décalage droite d'un octet pour obtenir le mot de 32 bits pour les pixels X[i][j+1]
- Instruction SIMD de minimum sur des octets non signés.

Définir le code VHDL pour l'implantation de ces différentes instructions et vérifier le fonctionnement correct par simulation logique.

2. Gauss

Sur un tableau N x N pixels (niveau de gris), on applique le filtre de Gauss 3 x 3 ci-dessous

$$1/16 * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

On veut réaliser les opérateurs matériels pour des instructions SIMD sur 32 bits permettant d'accélérer l'application du filtre de Gauss.
Les instructions à définir sont

- Décalages à gauche et droite d'un octet de mots de 32 bits (déjà définis dans la question précédente)
- Décompactage de deux octets bas en deux demi-mots de 16 bits (extension de zéros)
- Décompactage de deux octets haut en deux demi-mots de 16 bits (extension de zéros)
- Décalage gauche d'une et deux positions des demi-mots d'un mot de 32 bits
- Addition SIMD de deux demi-mots non signés.
- Compactage de deux demi-mots d'un premier mot en deux octets bas du mot résultat et des deux octets haut d'un deuxième mot en deux octets hauts du mot résultat. On considérera deux versions : la première considère que les demi-mots (16 bits) ont leur octet haut nul ; la seconde version sature l'octet bas lorsque l'octet haut du demi-mot est différent de 0.

Définir le code VHDL pour ces différentes instructions et vérifier le fonctionnement correct par simulation logique.

3. Filtre médian

Définir le code VHDL pour les instructions nécessaires à l'implantation du filtre médian et vérifier le fonctionnement correct par simulation logique.

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 Vizard*, 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 correspondant. 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 la plupart des opérateurs sont fournis (site Web de l'option).
Les seuls codes non fournis concernent les opérateurs de compactage.