Evaluation du calcul de racine carrée à l'aide d'un Processeur NIOS II de Altera

L'objectif de ce TP est de comparer les performances de différents systèmes embarqués ayant pour but d'exécuter l'algorithme de calcul de racine carrée suivant :

```
Charger X;
V= 2^(2n-2);
Z=0;
Pour i= n-1 à 0 faire
        Z=Z+V;
        Si ( X-Z)>=0  alors //on a ajouté V dans Z directement avant
                X=X-Z;
                Z= Z+V;//on ajoute encore un V pour Z
        Sinon
                Z=Z-V; //restituer ce qu'on a ajouté sur Z
        Fin si
        Z=Z/2;
        V=V/4;
Fin pour
```

Figure 1 : Algorithme de racine carrée considéré

Afin de tester l'impact sur les performances des différents choix matériels effectués dans l'élaboration du système embarqué, vous devrez mettre en œuvre plusieurs systèmes à base de processeurs NIOS II de Altera.
La mise en œuvre de ce processeur s'effectuera sur des cartes pédagogiques DE1 et nécessitera différentes étapes successives faisant appel à plusieurs logiciels.

a) Configuration du processeur :
Sous Quartus (serveur de license : 27001@tp-clew), afin de configurer le processeur, il faut lancer l'outil Qsys ou SOPC Builder (Menu *Tools*) après avoir créé un nouveau projet. Dans la fenêtre qui s'ouvre, il faut ensuite ajouter les éléments constitutifs du système envisagé :

- de la mémoire (vous prévoirez systématiquement d'utiliser les trois types de mémoires disponibles : Onchip Memory (16384 octets – mots de 32 bits), SDRAM (Custom, nécessité d'utiliser une PLL fournie), SSRAM (modèle University Program)).
- un processeur nios : Configurer les adresses des vecteurs reset et d'interruption en utilisant la SSRAM et prévoir un module simple (niveau 1) de débogage JTAG.
- un module de communication série JTAG : garder la configuration par défaut
- un module d'identification sysid
- rajouter des périphériques supplémentaires en fonction de l'application : timer, ports d'entrée/sorties …

Une fois la configuration terminée, vérifier qu'il n'y a pas de conflit au niveau des adresses sélectionnées pour les différents éléments et lancer la génération du système (Generate).

b) Mise en œuvre du processeur :
Vous devez charger le fichier d'assignation (*DE1_config.qsf*) regroupant les informations sur les connexions de tous les périphériques de la carte en cliquant sur *Assignments>>Import Assignments* puis en sélectionnant le fichier de configuration, en cochant dans le menu avancé l'option *global assignments*.

Modifier ensuite dans le fichier top_nios_system.vhd la déclaration du processeur nios et connecter les différentes pattes du système aux périphériques de la carte. Il faudra notamment relier l'entrée clk du processeur à la patte CLOCK_50 et le reset du processeur à la patte KEY[0]. Relier aussi les éventuelles autres entrées/sorties prévues en fonction de l'application aux pattes correspondantes de la carte DE1. Ceci fait, il faut compiler le schéma puis programmer le FPGA.

c) Programmation du processeur :

Lancer ensuite le programme NIOS IDE. Créer un nouveau projet C/C++ en associant bien le fichier .sopcinfo correspondant au processeur généré précédemment. C'est ce dernier qui décrit toutes les fonctionnalités et la carte des adresses associées au processeur généré. Choisissez un projet de type « blank_project ». Le programme vous crée alors deux sous répertoires : l'un d'entre eux contiendra les logiciels (*nom*) et l'autre contient les informations sur le système (*nom_syslib*).

Vérifier que toutes les fonctionnalités souhaitées ont bien été configurées en cliquant sur *NiosII >> BSP Editor* qui vous permet notamment de préciser la fonction des temporisateurs présents ainsi que de sélectionner les endroits où sont stockées les données (.rwdata) et les instructions (.text) dans l'onglet *Linker Script*. A chaque fois que vous modifiez quelque chose dans cet éditeur il faut regénérer le BSP (*generate*).

Créer ensuite un nouveau fichier C et entrer le programme. Il faudra inclure tous les fichiers header nécessaires (« system.h » + les fichiers header pour certains périphériques). En cliquant à droite sur le répertoire logiciel, il faut ensuite cliquer sur *Build Project* puis sur *Run As >> Nios2 Hardware* pour pouvoir exécuter le programme.

La console en bas de l'application IDE est la sortie STDOUT par défaut. C'est ici que s'afficheront notamment les résultats de vos printf.

<u>Travail demandé</u> (Faire un nouveau projet sous Quartus (nouveau répertoire) à chaque fois que vous devez changer la composition du système embarqué à l'aide de Qsys - normalement pour chaque partie**) :**

1) Prise en main des outils - utilisation de ports d'entrée/sortie :

a) Faire afficher par le processeur les valeurs des 16 Interrupteurs SW[15..0] sur les leds rouges LEDR[15..0]. Pour cela, il faudra prévoir au niveau du processeur, en plus des éléments décrits précédemment, deux ports d'entrées/sorties (PIO), un configuré en entrée, l'autre en sortie. Prévoir aussi un troisième PIO de 1 bit configuré en entrée pour indiquer au processeur qu'une nouvelle donnée est à récupérer. Le programme principal devra surveiller ce port d'entrée et afficher immédiatement les nouvelles données dès qu'elles sont présentes.

Le processeur NIOS utilisé devra être de type fast. Tester votre programme avec un processeur sans cache puis avec un processeur avec cache. Que constatez-vous ?

b) Utiliser une interruption pour déclencher la lecture des nouvelles données.

2) Exécution purement logicielle du calcul de racine carrée :

Afin de pouvoir estimer le temps de calcul du système, il faut inclure un temporisateur dans celui-ci, en le configurant pour qu'il ait une période assez grande. Ce dernier permettra de compter le nombre de coups d'horloge nécessaires au calcul des racines carrées. Pour avoir une meilleure visibilité des performances, le calcul devra se faire sur un grand nombre de valeurs différentes. Vous créerez donc un tableau d'entiers positifs de 200 cases dont le contenu est le carré de l'indice de la case. Les performances seront estimées en calculant le temps nécessaire pour effectuer 100 fois le calcul sur les 200 cases. Les résultats seront stockés dans un deuxième tableau qui sera remis à zéro à chaque nouvel essai.

a) Dans un premier temps, écrire une fonction qui permet de récupérer le contenu du registre du timer. La fonction *alt_timestamp_start()* permet de lancer le timer. Comparer le nombre de coups d'horloge nécessaires à l'appel de cette fonction et ceux nécessaires pour la fonction *alt_timestamp()* fournie par altera. Utiliser par la suite la plus rapide des deux solutions.
b) Exécuter l'algorithme de calcul de racine carrée sur l'ensemble du tableau soit directement depuis le programme principal, soit en faisant appel à une fonction (sqrt) pour chaque calcul et comparer les performances mesurées. Faire des estimations sur les trois versions du processeur en changeant pour chacune d'entre elles les types de mémoire utilisés pour les instructions et les données. Lorsque c'est possible, tester avec ou sans mémoire cache d'instruction ou de données. Vous devriez ainsi pouvoir remplir le tableau ci-dessous.

	Nios e	Nios s avec cache instr (512)	Nios s avec cache instr (2k)	Nios f avec cache instr + data* (512)	Nios f avec cache instr + data* (2k)
data + instr Onchip					
data + instr SSRAM					
data + instr SDRAM					

Tableau 1 : Performances comparatives des différentes architectures considérées - * : dans les deux derniers cas, la mémoire cache de donnée doit être mise en register et pas en blocs mémoire M4K

Donner pour chaque case le temps de calcul ramené à un calcul (en divisant par 20000 le temps affiché dans la console) de la version avec ou sans appel de fonction. Déterminer quelle est la meilleure configuration et le plus petit temps de calcul moyen pour une racine carrée. Donner à chaque fois aussi le nombre d'éléments logiques et de cellules mémoire utilisés (résultats de compilation).

3) Utilisation d'un coprocesseur :
Nous allons maintenant évaluer l'intérêt d'utiliser un coprocesseur associé au processeur Nios. Pour cela, nous allons implanter le module VHDL de calcul de racine carrée développé par vos soins dans le cadre des TPs VHDL. Les données ainsi que le signal de début seront envoyés par le processeur à travers des ports d'entrée/sorties et le résultat ainsi que le bit fin seront lus par le processeur. Vous prendrez pour ce dernier la version la plus rapide trouvée au 2).
a) Estimer le temps de calcul si le processeur surveille le bit fin dans le programme principal avant d'écrire le résultat dans le tableau.
b) *Optionnel (si les toutes les autres parties ont été traitées)* - Estimer le temps de calcul si le bit fin déclenche une interruption pour permettre d'écrire le résultat dans le tableau. Conclusions.

4) Utilisation d'instructions spécifiques :
Au lieu d'utiliser un coprocesseur, créer une instruction personnalisable pour inclure l'architecture de calcul de racine carrée décrite en VHDL à l'intérieur de l'ALU du processeur Nios. Il s'agira d'une instruction personnalisable nécessitant plusieurs coups d'horloge pour être exécutée. Comparer les performances obtenues avec les versions précédentes.

5) Utilisation d'un composant directement associé au Nios :
L'idée est ici de modifier le code VHDL de l'architecture de calcul de racine carrée afin de pouvoir l'associer directement au processeur NIOS à l'intérieur de Qsys, sans passer par des ports d'entrées/sorties. Comparer les performances obtenues avec les versions précédentes.