# TP1 Architecture des processeurs

Ce TP permet une initiation à la conception de systèmes numériques assistée par ordinateur. Il vous est demandé de résoudre un problème de conception numérique, d'utiliser pour cela le langage VHDL et de simuler le système numérique pour valider le fonctionnement. Le développement des solutions sera fait en VHDL sur la base de l'outil de simulation MODELSIM. Nous implanterons ensuite ces solutions sur une carte FPGA. Nous utiliserons les outils QUARTUS de Intel FPGA (ex ALTERA) et les cartes DE0 ou DE1.

## Exercice 1 :

On suppose N (N=16) données rangées dans un tableau. On veut réaliser un automate recherchant dans ce tableau la présence d'un code (sur 8 bits). L'automate renvoie en sortie, une adresse du tableau donnant la position du code recherché ou alors un signal NON_EXIST indiquant l'échec de la recherche.

1) On suppose les données du tableau rangées dans un ordre quelconque. Décrire l'algorithme puis l'implanter en VHDL dans une approche comportementale.

2) Réaliser la simulation et évaluer les temps de calcul en fonction du nombre de cycle d'horloges.

## Exercice 2 :

On désire réaliser un multiplieur basé sur l'algorithme des additions décalages. Ce multiplieur prend deux valeurs en entrée A et B sur 8 bits non signés et donne un résultat R sur 16 bits. Les deux valeurs A et B sont mémorisées dans deux registres. L'algorithme de multiplication commence par le bit de poids faible du multiplicateur B.
L'addition au résultat temporaire sera réalisée sur 8 bits, sauvée sur Rh (Poids fort de R) et le décalage du registre R est réalisé vers la droite. L'algorithme est le suivant :

On considère que R est composé de deux registres Rh et Rl.
Les additions s'effectuent sur le registre Rh.

*A<- entrée de la valeur du multiplicande;*
*B<- entrée du multiplicateur*
*R=0;  /* registre résultat */*
*pour i =0 à 7 faire*
    *Rh=Rh+A et $b_0$; /* et logique bit à bit entre $b_0$ et A noté (a7… a0) */*
    *shift right R;*
    *shift right B;*
*fin pour*

Pour réaliser ce multiplieur, il faut implanter des fonctions de base tels que des registres à décalage, additionneur 8 bits (avec une retenue sortante) ...
Il y a plusieurs approches possibles pour implanter cet opérateur. Nous adopterons une approche dite comportementale. Dans ce cas toute l'architecture est décrite dans un seul process (VHDL).

**1)** Après avoir précisé sur feuille les détails des opérations à effectuer ainsi que leur séquencement, décrire le process permettant de réaliser cette opération.

**2)** Simuler ce système sous MODELSIM. On prendra soin d'ajouter les signaux d'interface avec l'extérieur, à savoir, le signal de « start » et « fin » pour lancer le calcul et attendre la fin du résultat.

**3)** Implanter cet opérateur sur la carte DE1. Il est recommandé de bien lire le support d'aide du TP. On utilisera les interrupteurs (SW[7..0]) pour entrer les données A et B. L'interrupteur SW[8] permet de choisir quel registre d'entrée on mémorise (A ou B). Enfin, l'interrupteur SW[9] » permet de lancer le calcul de cette multiplication et la LED[0] permet d'indiquer la fin du calcul. La fréquence de l'Horloge sera de 50 MHz.

# TP 2&3 Architecture des processeurs

On se propose de réaliser un circuit pour calculer la racine carrée entière par défaut d'un nombre X sur 16 bits (c.à.d. le plus grand entier dont le carré soit inférieur ou égal à X). Cet algorithme est donné ci-dessous :

```
Charger X;
V= 2^(2n-2);
Z=0;
Pour i= n-1 à 0 faire
        Tmp=X-V;
        Tmp=Tmp –Z;
        Z=Z/2;
        Si (TMP>=0) alors
                X= TMP;
                Z= Z+V;
        Fin si
        V=V/4;
Fin pour
```

Pour simplifier la mise en oeuvre de l'algorithme et éviter de travailler en signé (à cause de la soustraction sur TMP), nous mettrons en oeuvre une version modifié suivant :

```
Charger X;
V= 2^(2n-2);
Z=0;
Pour i= n-1 à 0 faire
        Z=Z+V;
        Si ( X-Z)>=0  alors
                X=X-Z;
                Z= Z+V;
        Sinon
                Z=Z-V;
        Fin si
        Z=Z/2;
        V=V/4;
Fin pour
```

Dans cette nouvelle version de l'algorithme, l'opération X= X-Z ne sera effectuée que si X-Z est positif. D'autre part, l'opération sur Z (voir correction du TD) est Z = (Z+2V)/2. Dans l'algorithme 1, on considère le cas idéal mais en réalité le fait d'effectuer l'opération Z= Z/2 + V fait que nous perdons en précision dans le cas d'un calcul entier. L'algorithme 2 effectue alors l'opération Z=Z+V en premier, pour pouvoir utiliser ce résultat pour calculer la nouvelle valeur de X = X-Z-V. Avec le calcul préalable Z=Z+V, l'opération sur X devient X=X-Z (la valeur V est incluse dans Z).
Il y a plusieurs approches possibles pour implanter cet opérateur. Nous adopterons une approche dite comportementale. Dans ce cas toute l'architecture est décrite dans un seul process (en VHDL).

**1)** Après avoir précisé sur feuille les détails des opérations à effectuer ainsi que leur déroulement, décrire le process permettant de réaliser cette opération de calcul.

**2)** Simuler ce système sur MODELSIM.

**3)** Implanter cet opérateur sur la carte DE1 (mise à votre disposition). On utilisera les interrupteurs (SW[7..0]) pour entrer la donnée sur 16 bits. L'interrupteur SW[8] sera utilisé pour charger la partie haute (Xh) et la partie basse (Xl) de la donnée X. Enfin, l'interrupteur SW[9] (start) sera utilisé pour lancer le calcul de la racine carrée. Le bouton « KEY[0] » permettra d'initialiser le système (reset). La fréquence de l'Horloge sera de 50 MHz. On visualisera le résultat sur les afficheurs 7-segments.

# Support TP Architecture des processeurs

## Introduction :

Le but des TDs/TPs VHDL est évidement de vous donner des notions plus concrètes de ce langage et de vous faire découvrir les différents niveaux de description des architectures des systèmes. Le langage VHDL est très utilisé dans le monde industriel car, contrairement à une description schématique très dépendante des outils, cette description est textuelle. Cela permet alors d'utiliser cette même description textuelle sur des cibles physique différentes (XILINX, EPLD, ASIC, composants sur carte...).
La conception d'une architecture utilisant VHDL doit respecter un cheminement logique suivant :

- Description textuelle à l'aide d'un éditeur de textes classiques (ASCII, ne pas utiliser des traitements de textes, fichiers binaires).
- Compilation de cette description afin de déceler des erreurs de syntaxe.
- Simulation fonctionnelle pour valider le comportement du système.
- Choix d'une cible d'implantation et synthèse du fichier vers cette cible.
- Placement et routage de la description structurelle relative à la cible choisie.
- Simulation temporelle afin de vérifier les problèmes de délais et les fréquences de fonctionnement.
- Programmation ou réalisation du système.

Les séances de TPs vont concerner essentiellement les trois premières phases décrites précédemment, sachant que la phase la plus importante sera donc de maîtriser la syntaxe et d'apprendre certaines subtilités de ce langage à travers des exemples.
L'outil utilisé est un compilateur/simulateur VHDL V-SYSTEM de Model Technology (Mentor Graphics, Leader des outils CAO), sous WINDOWS. Le cœur de l'outil est implanté sur d'autres outils de CAO.

## Démarche :

Après avoir édité un fichier VHDL (avec une extension .VHD obligatoire) sous l'éditeur NOTEPAD de WINDOWS ou PFE (ou autre), lancez l'outil **Modelsim** (explications données en TP). Une explication détaillée de l'utilisation de cet outil de compilation/simulation vous sera donnée pendant le TP.

- *.....Explications sur l'utilisation du VHDL et de l'outil sur WINDOWS*

## AIDE au démarrage de MODELSIM

Après avoir lancé l'outil, il faut obligatoirement changer le répertoire de travail vers le votre. Cette opération sera réalisée en allant dans le menu ***FILE→Change DIRECTORY*** de l'outil.

Enfin, si c'est la première fois que vous travaillez dans ce répertoire, il faudra obligatoirement créer un WORK directory (un répertoire lib de travail de V-system). Créer cette librairie work dans le menu ***FILE→LIBRARY***. Le nom WORK est automatique inséré dans ce menu.

**ATTENTION : Cette opération ne doit être faite qu'une fois par répertoire de travail.**

## AIDE mémoire pour la simulation

**Add wave ss** permet d'ajouter le signal ss dans la fenêtre de timing (/*: tous les signaux).
**force cc 0 0, 1 50 -repeat 100** permet de créer une horloge **cc** de période 100ns et de niveau 1 de 50ns avec un rapport cyclique 50%.
**run xx** lance la simulation pendant **xx** ns.
**dofile fichier** permet de lancer un fichier script contenant des ordres de simulation.

## AIDE mémoire pour VHDL :

(**V**ery high speed integrated circuits **H**ardware **D**escription **L**angage)

Un système est souvent décrit sous forme d'un schéma. Ce dernier regroupe l'ensemble des composants utilisés. Ces composants sont des symboles graphiques (une boite avec des signaux d'E/S). Le contenu d'un symbole de type composant sera également un schéma qui rassemble des composants plus simples. On a alors une hiérarchie entre les différentes feuilles schématiques du système ainsi décrit.
Dans le cas de VHDL un schéma est considéré comme une description textuelle d'une fonctionnalité. Le bloc ARCHITECTURE permet de décrire le comportement interne de votre système (sous forme d'équations logiques ou bien autres…) la syntaxe sera :

```
-- ceci est un commentaire
ARCHITECTURE  archi1 OF symbole1 is
--ici seront insérés tous les signaux utilisés
signal x : bit; --signal x de type bit interne à l'architecture
BEGIN
        --ici seront insérées les équations et les descriptions fonctionnelles
END archi1;
```

Ce système, ainsi décrit, sera visible de l'extérieur comme une boite noire avec des signaux d'entrées sorties (signaux d'échanges avec l'extérieur). La définition de ces signaux d'échanges est nécessaire. Tous les signaux décrits dans cette définition sont visibles de l'extérieur et à l'intérieur de l'architecture du système. Cette description est désignée par ENTITY dont la syntaxe sera :

```
ENTITY symbole1 is
        PORT (
-- ici déclaration
            signal1 : IN bit;
            signal2  : OUT bit;
        );
END symbole1;
```

Nous avons dans cet exemple décrit 2 signaux d'échanges de type bit (1 bit).